Optimization of Plunger Lift Performance in Stripper Gas Wells

during the Period 05/15/2001 to 11/30/2002

By

Erdal Ozkan
Colorado School of Mines

March 2003

Work Performed Under Prime Award No. DE-FC26-00NT41025
Subcontract No. 2036-CSM-DOE-1025

For
U.S. Department of Energy
National Energy Technology Laboratory
P.O. Box 10940
Pittsburgh, Pennsylvania 15236

By
Colorado School of Mines
1500 Illinois Street
Golden, CO 80401

DE-FC26-00NT41025

OPTIMIZATION OF PLUNGER LIFT PERFORMANCE IN STRIPPER GAS WELLS

Final Report
May 15, 2001 - December 14, 2002

Principal Author
Erdal Ozkan

Submitting Organization:
Colorado School of Mines
1500 Illinois Street
Golden, CO 80401

March, 2003

DISCLAIMER

ABSTRACT

In this project, an algorithm has been developed and tested to optimize the production and shut-in periods of plunger-lift operation based on reservoir performance. The objective of the optimization is to maximize gas production with the condition that the liquid loaded during production can be lifted to surface by the pressure that builds up during the following shut-in period. The optimization of the production and shut-in periods simultaneously requires an iterative procedure. One of the advantages of the proposed optimization method is the ability to automatically adjust to the changes in the line pressure.

The optimization algorithm combines the conventional plunger-lift theory with an analytical description of the reservoir performance. The conventional plunger-lift theory is used to determine the pressures required for lifting the plunger with a liquid column over it. The production and shut-in times are determined in an iterative manner by using an analytical reservoir model to simulate the reservoir performance.

To implement the algorithm, a relatively simple electronic control system has been designed and manufactured. The cost of the control system is under $1,000 and may be connected to the existing wellhead controls with minimal modification. Using computer simulations, the method has been tested with the field data and indicated that up to 100% increase in the cumulative production may be achieved by optimizing the production and shut-in periods based on the reservoir performance. Three field application tests failed because of what appeared to be power supply related problems. Field testing will resume with stricter regulations on power supply when the winter conditions at the test-well site improve.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Plunger lift is one of the viable options to produce low volume, stripper gas wells. The efficiency of the plunger-lift production, however, strongly depends on the regulation of the production and shut-in periods. Various techniques have been developed to determine the durations of the production and shut-in periods but they do not use the reservoir performance as their bases. The objective of this project was to develop an algorithm that could optimize the production and shut-in periods of plunger-lift operation based on reservoir performance.

The optimization algorithm uses an iterative approach to determine the production and shut-in periods. The shut-in period depends on how much liquid builds up in the wellbore during the production period and is desired to be minimum. The production period, on the other hand, should be maximized with the requirement that the reservoir be able to build the pressure during the following shut-in period to the level required to lift the produced liquid. The optimization algorithm uses the conventional plunger-lift theory to determine the pressures required for lifting the plunger with the liquid column over it. The production and shut-in times are determined by using an analytical reservoir model in an iterative manner.

The optimization algorithm developed in this project has been tested with the field data in computer simulations. The results indicate that up to 100% increase in the cumulative production may be achieved by optimizing the production and shut-in periods considering the reservoir performance. One of the important advantages of the proposed algorithm is the ability to automatically adjust to the changes in the line pressure. The implementation of the algorithm requires a relatively simple electronic control system that may be built under $1,000 and can be connected to the existing wellhead controls with minimal modification. Three application tests in the field became unsuccessful with technical problems that do not seem to be related to the optimization algorithm and electronic control system developed in this project. Further testing are planned under improved technical conditions when the weather conditions permit.

INTRODUCTION

Low volume stripper wells account for 8 % of the natural gas production in the United States. A stripper well is defined as a well that produces 60 MSCFD of natural gas or less. As the price of gas and oil declines, many of these wells are abandoned because the production and maintenance costs are higher than the selling price. To improve the profitability of these wells, the production needs to be optimized. One of the problems faced by these wells is the production of liquid. Stripper gas wells are loaded periodically with liquid (either condensate or water) and has to be unloaded before the production can begin again. One common solution to unload is the use of plunger lift. The efficiency of production with plunger lift, however, depends strongly on the durations of the production and shut-in periods.

During the shut-in period of plunger-lift production, the plunger sits at the bottom of the tubing with a liquid column resting above it (Fig. 1). The well is shut in for a period until the casing pressure is high enough to lift the plunger. The plunger is lifted with high-pressure gas and, along with liquid, gas is produced until the gas flow rate starts decreasing again. During the production, liquid will continue to load in the well. The well is shut in again, the plunger will drop at the bottom, and the cycle will continue. The plunger has a one way valve in the middle so that the liquid can move through it while the plunger is dropping, but liquid cannot drop while the plunger is lifted to the surface.

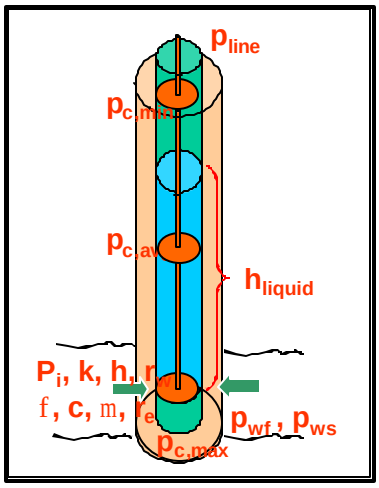

Fig. 1 – Schematic of a plunger at the bottom of the well with a liquid column above it.

An important parameter determining the efficiency of the production-lift performance is the duration of the production and shut-in periods. The methods used to determine the time periods, in general, are arbitrary and do not directly take into account the performance of the reservoir. One of the common procedures is to determine the time periods by trial and error and then put the plunger on a timer clock with predetermined

production and shut-in periods. In the alternative, the flow rate is monitored and when the gas flow falls below a threshold rate, the well is shut in.

In this project, an optimization algorithm for the production and shut-in periods of plunger lift has been developed and implemented. This procedure is based on the reservoir performance and may accommodate the changes in the line pressure. The following activities have been undertaken and completed during the project:

Activity 1 - Development of a Production Optimization Algorithm: The development of the optimization algorithm was the main objective of the project. This activity included two specific tasks:

> ***Task 1.1 - Determination of Reservoir Properties:*** This task dealt with the necessity that the reservoir properties, such as permeability, porosity, skin, etc., must be known to condition the plunger lift operation to the reservoir performance. Normally, reservoir properties are determined by pressure-transient analysis but, because of cost considerations, stripper gas wells usually lack pressure transient data. A practical solution to this problem may be the use of production data. Therefore, investigating the potential of using production data to determine reservoir properties was one of the tasks of this project.
>
> ***Task 1.2 - Development of Production Optimization Algorithm:*** The main objective of this project was to optimize the plunger lift performance based on reservoir performance. This objective required simulating the reservoir performance to determine the optimum production and shut-in intervals so as to maximize the production from plunger lift. Particular emphasis has been given to the development of a robust algorithm that only required average reservoir properties and could be easily implemented on the existing well controls.

Activity 2 - Field-Testing and Validation of the Proposed Method: The algorithm and the method have been validated with the field data and implemented on a well. Three tasks have been involved in this activity:

> ***Task 2.1 – Building the Electronic Box:*** The implementation of the optimization method required manufacturing an electronic box, which included the production optimization algorithm and the switch controls.
>
> ***Task 2.2 – Field-Testing of the Method:*** The optimization algorithm and the electronic box have been tested by using the field data first and then by applying on a gas well produced by plunger lift.
>
> ***Task 2.3 – Final Report:*** Throughout the project, the results have been compiled to construct the final report to be presented to the Stripper Well Consortium.

The details of the above activities and the results obtained from the applications are reported in the following sections of this report.

EXPERIMENTAL

The proposed method of this research project was semi-analytical. No experimental task has been performed.

RESULTS AND DISCUSSION

The results of the project are documented and discussed below with respect to the specific activities and tasks proposed originally and documented in the Introduction. The details of the developments are provided in the Appendices.

Activity 1 - Development of a Production Optimization Algorithm:

Task 1.1 - Determination of Reservoir Properties:

The objective of this task was to investigate the possibility of using production data to obtain the reservoir properties required by the optimization algorithm. We have extensively investigated the methods proposed in the literature[1-3] to estimate the average reservoir properties by using the production history and proposed an extension of the existing methods. Below, we present a summary of our research on the determination of reservoir properties and original gas in place from production data. Additional details are given in Refs. 4 and 5 and Appendix A.

We first introduce the concept of dimensionless transient productivity index. We show that the dependence of the dimensionless transient productivity index on the bottomhole production conditions is a weak one and may be neglected for practical purposes of liquid production. This allows us to use the same set of type curves for constant and variable rate/pressure conditions at the bottomhole. As in the works of Palacio and Blasingame[1] and Agarwal et al.[2] We, then, extend these ideas to gas production conditions by using pseudopressure and pseudotime concepts discussed in the literature.[1,2]

Dimensionless Transient Productivity Index, J_D: Productivity index, J, is a conventional definition of a well's productivity under stabilized flow conditions and is defined by

$$J = \frac{q}{\overline{p} - p_{wf}}, \tag{1}$$

where $\overline{p}$ represents the average reservoir pressure. For liquid wells and gas wells under Darcy flow conditions, the productivity index, *J*, is a constant and independent of the bottomhole flow conditions.[6]

In developing type-curves for the analysis of transient pressure responses, it is customary to define dimensionless variables as follows: The dimensionless bottomhole pressure based on the initial pressure, p_i, is defined by

$$p_{wD}(t_D) = \frac{kh}{141.2qB\mu}\left[p_i - p_{wf}(t)\right], \tag{2}$$

where q may be constant for constant-rate production or a function of time [$q = q(t)$] for variable-rate production.

It is possible to define a dimensionless bottomhole production rate by

$$q_D(t_D) = \frac{141.2q(t)B\mu}{kh\left(p_i - p_{wf}\right)}, \tag{3}$$

where p_{wf} may be a fixed bottomhole pressure for production at a constant pressure or may change as a function of time [$p_{wf} = p_{wf}(t)$]. The dimensionless time is defined by

$$t_D = \frac{2.637\times10^{-4}kt}{\phi c_t \mu r_w^2}. \tag{4}$$

For our purposes, it is also possible to define a dimensionless bottomhole pressure based on average pressure, $\bar{p}$, as follows:

$$\tilde{p}_{wD}(t_D) = \frac{kh}{141.2qB\mu}\left[\bar{p} - p_{wf}(t)\right] = p_{wD} - \bar{p}_D, \tag{5}$$

where $\bar{p}_D$ is the dimensionless average pressure given by[6]

$$\bar{p}_D = \frac{kh}{141.2qB\mu}\left[p_i - \bar{p}(t)\right] = 2\pi t_{AD}, \tag{6}$$

and t_{AD} is a dimensionless time defined based on the drainage area, A, as follows

$$t_{AD} = \frac{2.637\times10^{-4}kt}{\phi c_t \mu A} = \frac{r_w^2}{A}t_D. \tag{7}$$

Similarly, we can define a dimensionless rate based on average pressure as follows:

$$\tilde{q}_D(t_{eD}) = \frac{141.2q(t_e)B\mu}{kh\left[\bar{p} - p_{wf}(t_e)\right]} = \frac{q_D(t_{eD})}{1 - \bar{p}_{Dcp}(t_{eD})} = \frac{1}{1/q_D(t_{eD}) - \bar{p}_D(t_{eAD})}. \tag{8}$$

In Eq. 8, $\overline{p}_{Dcp}$ is the dimensionless average pressure for constant pressure production conditions given by[6]

$$\overline{p}_{Dcp} = \frac{p_i - \overline{p}(t_e)}{p_i - p_{wf}} = \frac{2pQ_D(t_{eD})}{A/r_w^2}, \tag{9}$$

where

$$\begin{aligned} Q_D(t_{eD}) &= \int_0^{t_{eD}} q_D(t)dt = \frac{3.723\times10^{-2}Q(t_e)B}{khfc_t r_w^2(p_i - p_{wf})} \\ &= \frac{3.723\times10^{-2}B}{khfc_t r_w^2(p_i - p_{wf})}\int_0^{t_e} q(t)dt \end{aligned}. \tag{10}$$

The last equality in Eq. 10 follows from the definition of average pressure $\overline{p}_D(t_{eAD})$ for general variable rate production conditions as follows:[6]

$$\overline{p}_D(t_{eAD}) = \frac{kh}{141.2q(t_e)Bm}[p_i - \overline{p}(t_e)] = 2pt_{eAD}, \tag{11}$$

where

$$t_e = \frac{1}{q(t)}\int_0^t q(t)dt = \frac{Q(t)}{q(t)}. \tag{12}$$

Note that t_e defined in Eq. 12 is the same as the material balance time used by Palacio and Blasingame.[1] Also note that for constant pressure production, $t_e = t$. Therefore, in the following discussions, we will use t_e for both constant rate and variable rate production conditions.

If we define a dimensionless transient productivity index, $J_D(t_D)$, as follows

$$J_D(t_{eD}) = \frac{141.2Bm}{kh}J(t_e), \tag{13}$$

where

$$J(t_e) = \frac{q(t_e)}{\overline{p} - p_{wf}(t_e)}, \tag{14}$$

then, Eqs. 5, 6, 13, and 14 may be combined to yield

$$J_D(t_{eD}) = \frac{1}{\tilde{p}_{wD}(t_{eD})} \qquad \text{for constant rate production,} \qquad (15)$$

and

$$J_D(t_{eD}) = \tilde{q}_D(t_{eD}) \qquad \text{for variable rate production.} \qquad (16)$$

We may now note the following features of the dimensionless transient productivity index: During transient flow periods, because $\overline{p} \approx p_i$, we have

$$J_D(t_{eD}) = \frac{1}{p_{wD}(t_{eD})} \qquad (17)$$

or

$$J_D(t_{eD}) = q_D(t_{eD}), \qquad (18)$$

(depending on the bottomhole flow conditions) where $p_{wD}(t_D)$ and $q_D(t_D)$ represent the conventional dimensionless pressure and rate defined by Eqs. 2 and 3, respectively. This implies that the transient flow portions of the theoretical dimensionless transient productivity-index curves may be constructed by using the conventional type curves used for well test analysis.

At late times, when the boundary effects dominate the well response, $J(t_e)$ term defined in Eq. 14 becomes a constant (assuming Darcy flow conditions) equal to J defined in Eq. 1. Therefore, the boundary dominated flow portion of the dimensionless transient productivity index curve is a constant that is proportional to the inflow performance relationship for the given well/reservoir system. Figure 2 shows a production decline type curve for a vertical well in a closed circular reservoir. Each curve on the type curve corresponds to a specific value of dimensionless drainage radius, $r_{eD} = r_e / r_w$, noted on the figure. The curves shown on Fig. 2 have been generated by numerically inverting the known analytical solutions in the Laplace domain (see Appendix A).

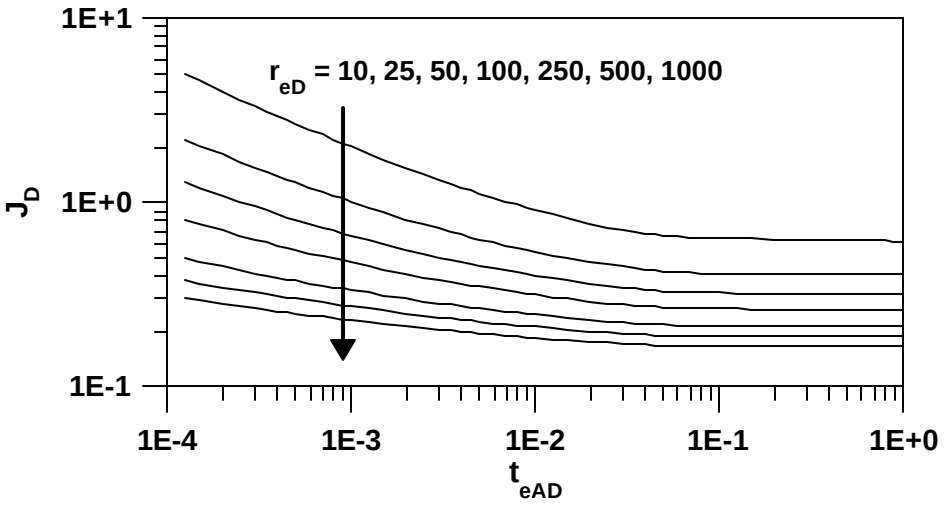

Fig. 2 – Production decline type curve in terms of dimensionless productivity index for a vertical well in a closed circular reservoir.

Effect of Mode of Production on J_D: As discussed by Palacio and Blasingame[1] and Agarwal et al.,[2] $q_D(t_{eD})$ vs. t_{eD} for constant bottomhole production (BHP) should follow the constant rate production (CRP) responses plotted in terms of $1/p_{wD}(t_{eD})$ versus t_{eD}. Here, we apply these ideas to dimensionless productivity index and investigate the equivalence of the transient productivity indices for constant rate and constant pressure production conditions; that is, we investigate the possibility that $\tilde{q}_D(t_{eD})$ versus t_{eD} curves follow $1/\tilde{p}_{wD}(t_{eD})$ versus t_{eD}, where $\tilde{q}_D$ and $\tilde{p}_{wD}$ are defined by Eqs. 5 and 8, respectively.

For a vertical well in a bounded reservoir, it is a well-known result that $q_D(t_D) \approx 1/p_{wD}(t_D)$ during transient flow period. Because during transient flow, $q_D(t_D) \approx \tilde{q}(t_D)$ and $p_{wD}(t_D) \approx \tilde{p}_{wD}(t_D)$, we can expect to have $\tilde{q}_D(t_{eD})$ versus t_{eD} curves for constant pressure production to follow $1/\tilde{p}_{wD}(t_{eD})$ versus t_{eD} curves for constant rate production. Therefore, the dimensionless productivity index, J_D, should be approximately independent of wellbore production conditions during transient flow period. Also, at late times (boundary dominated flow), $J = q/(\overline{p} - p_{wf})$ is approximately independent from the production conditions,[6] which by Eqs. 15 and 16 implies that the dimensionless productivity index, J_D, should also be independent of the production conditions at late times. Then, we may expect to have $J_D(t_{eD})$ to be approximately independent of the production conditions for all times.

Figure 3 compares the dimensionless productivity indices of a vertical well in a closed cylindrical reservoir of dimensionless radius, $r_{eD} = r_e/r_w = 20$ under constant-rate production (the unbroken line in Fig. 3) and constant-pressure production (the circular data points) conditions.[4,5] These results were generated by numerically inverting

the analytical solutions in the Laplace domain (see Appendix A). The agreement of the results during the late times (when the dimensionless productivity index becomes a constant) is satisfactory. During the transient flow period, the results for the constant-pressure production case are slightly below the results for the constant-rate production case, but the agreement is acceptable for practical purposes. Also for comparison, Fig. 3 shows $1/p_{wD}$ and q_D vs. t_{eD} (the dashed line and the square data points, respectively) as suggested by Refs. 1 and 2. The agreement between the constant pressure and constant rate production results is similar to that observed for the transient productivity index results.

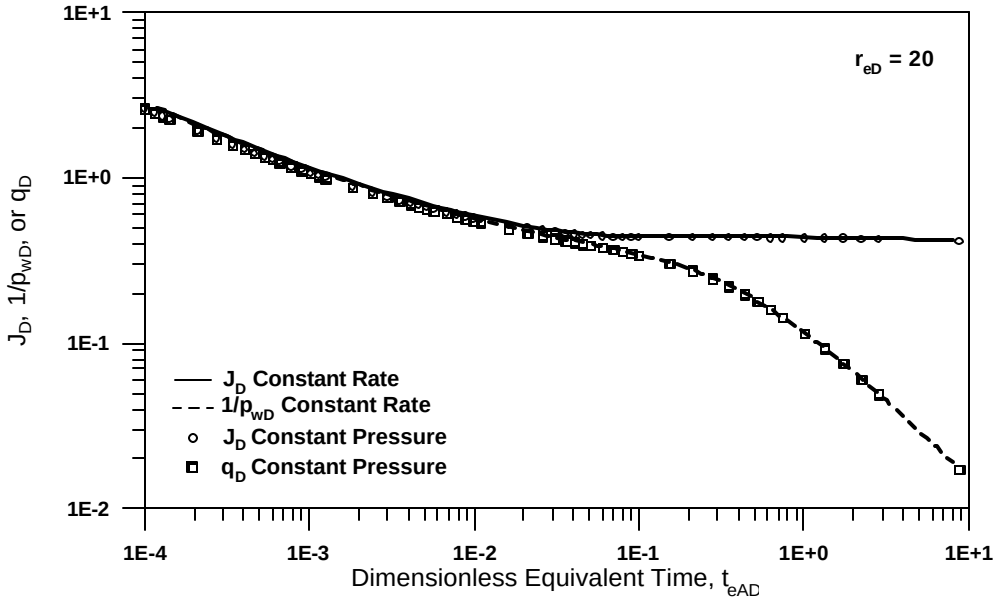

Fig. 3 – Comparison of production decline type curves for constant rate and constant pressure production conditions.[4,5]

Extension to Gas Reservoirs: The ideas developed above may be extended to gas reservoirs by using the pseudopressure and pseudoequivalent time[1,2] defined, respectively, by

$$m(p) = 2\int_{p_0}^{p} \frac{p'}{m\,Z}\,dp', \tag{19}$$

where p_0 is an arbitrary datum pressure, and

$$t_a = \frac{m_i c_{ti}}{q(t)} \int_0^t \frac{q}{m(\bar{p}) c_t(\bar{p})} dt$$
$$= \frac{m_i c_{ti}}{q(t)} \frac{Z_i G}{2 p_i} \left[m(\bar{p}) - m(p_{wf}) \right] \tag{20}$$

where G is the gas in place and the subscript i indicates the initial conditions. Note that the computation of the pseudoequivalent time given in Eq. 20 requires estimates of fluid properties as a function of the average reservoir pressure. The average reservoir pressure profile may be estimated from an estimate of gas in place, G, by using the gas material balance equation given by

$$\frac{1}{G} = \frac{1}{G_p} \left[1 - \frac{\bar{p}}{\bar{Z}} \frac{Z_i}{p_i} \right]. \tag{21}$$

If the gas in place is not known, the iterative procedure discussed below in the Analysis Technique section may be used to obtain an estimate of G.

The dimensionless versions of the pseudopressure based on the initial and average pressures are given, respectively, by

$$m_{wD}(t_{aD}) = \frac{kh}{1422Tq} \left[m(p_i) - m(p_{wf}) \right], \tag{22}$$

and

$$\tilde{m}_{wD}(t_{aD}) = \frac{kh}{1422Tq} \left[m(\bar{p}) - m(p_{wf}) \right]. \tag{23}$$

The dimensionless pseudoequivalent times based on the wellbore radius, r_w, and drainage area, A, are defined, respectively, by

$$t_{aD} = \frac{2.637 \times 10^{-4} k t_a}{f (m c_t)_i r_w^2}, \tag{24}$$

and

$$t_{aAD} = \frac{2.637 \times 10^{-4} k t_a}{f (m c_t)_i A}. \tag{25}$$

We also define the dimensionless production rate based on initial and average pressure as shown, respectively, below.

$$q_D(t_{aD}) = \frac{1422Tq(t_a)}{kh[m(p_i) - m(p_{wf})]}, \tag{26}$$

and

$$\tilde{q}_D(t_{aD}) = \frac{1422Tq(t_a)}{kh[m(\bar{p}) - m(p_{wf})]}. \tag{27}$$

We also use the definition of dimensionless cumulative production given below.

$$\begin{aligned} Q_{aD}(t_{aD}) &= q_D t_{aD} \\ &= \frac{9.0T}{fh[m(p_i) - m(p_{wf})]r_w^2} \int_0^t \frac{q(t)}{m(\bar{p})c_g(\bar{p})} dt \\ &= \frac{4.5TZ_iG}{fhr_w^2 p_i} \left[\frac{m(p_i) - m(\bar{p})}{m(p_i) - m(p_{wf})} \right] \end{aligned} \tag{28}$$

We can, now, note the following relations:

$$\tilde{m}_{wD}(t_{aD}) = m_{wD}(t_{aD}) - m_D(\bar{p}), \tag{29}$$

where

$$m_D(\bar{p}) = 2pt_{aAD}, \tag{30}$$

and

$$\tilde{q}_D(t_{aD}) = \frac{q_D(t_{aD})}{1 - m_{Dcp}(\bar{p})} = \frac{1}{1/q_D(t_{aD}) - m_D(\bar{p}_D)}, \tag{31}$$

where

$$m_{Dcp}(\bar{p}) = \frac{p_i - \bar{p}(t_a)}{p_i - p_{wf}} = \frac{2pQ_{aD}(t_{aD})}{A/r_w^2}, \tag{32}$$

and

$$m_D(\bar{p}_D) = 2pt_{aAD}. \tag{33}$$

Using the above relations, we may define the dimensionless productivity index for gas wells as follows:

$$J_D(t_{aD}) = \frac{kh}{1422T} J(t_a), \tag{34}$$

where

$$J(t_a) = \frac{q(t_a)}{m(\bar{p}) - m(p_{wf})}. \tag{35}$$

Because the gas well responses in terms of pseudopressure should follow the liquid well responses in terms of pressure, and the definition of the pseudoequivalent time should remove the dependency of the transient productivity index on the mode of production, we can expect to have the dimensionless transient productivity index for gas wells to follow that for liquid wells. Numerical proof of these ideas has been presented in Ref. 4.

Analysis Technique: The objective of the decline-type-curve analysis is to determine the reserves and reservoir properties. The analysis technique is similar to that proposed by Agarwal *et* al.[2] and uses the ideas presented by Palacio and Blasingame.[1]

First, an estimate of the original gas in place, *G*, should be obtained. As suggested by Agarwal *et* al.[2] a trial and error procedure may be used to estimate *G*. An initial guess may be chosen between the cumulative gas production (lower limit) and the volumetric estimate from petrophysical data (upper limit) to start iterations. It is shown in Fig. 4 that an underestimation of the *G* will cause an upward bend (increase) of the productivity index during boundary dominated flow, whereas an overestimation will cause a downward bend (decrease). The convergence to a correct estimate will be verified by a constant productivity index during the boundary dominated flow period. In this iteration process, convergence is usually obtained rapidly.

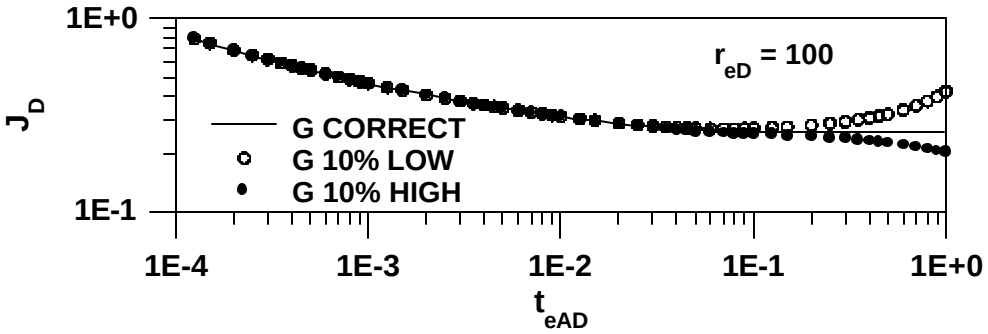

Fig. 4 – Iterative estimation of the gas in place, *G*.

After an accurate estimate of *G* is obtained, the analysis continues with type curve matching to estimate the permeability. The productivity index curve computed from the field data is matched with one of the type curves, for example, on Fig. 2. Once a reasonably good match is obtained, a match point (M.P.) is chosen and the corresponding

values of the dimensionless (type curve) and field (data) productivity indices and the dimensionless (type-curve) and dimensional (data) times are noted. The match point values are then used in the following equation to estimate the permeability:

$$k = \frac{1422T}{h} \frac{\left\{ \frac{q}{\left[m(\bar{p}) - m(p_{wf}) \right]} \right\}_{M.P.}}{\left[J_D \right]_{M.P.}}. \tag{36}$$

After estimating the permeability, k, an estimation of G may be obtained from the following equation:

$$G = \frac{6.3288 \cdot 10^{-3} kh}{5.615 m_i c_{ti} B_{gi}} \frac{\left[t \right]_{M.P.}}{\left[t_{AD} \right]_{M.P.}}. \tag{37}$$

The theoretical development presented above is independent of the type of well used in the production. The only minor modification required is the replacement of the wellbore radius, r_w, used in the dimensionless definitions by the half-length of the fracture and well for fractured and horizontal wells, respectively. References 4 and 5 present the application of these ideas to horizontal wells.

Application Example: Here, we consider the example application discussed in Refs. 4 and 5 for a horizontal gas well produced at a constant rate, $q = 8000$ MSCF/d. The properties of the horizontal well and reservoir are presented in Table 1 and the measured bottomhole pressure, p_{wf}, and the corresponding pseudopressures, $m(p)$, are shown in Fig. 5 as a function of time.

Table 1 – Reservoir and fluid data for the example application.[4]

A, area, Acres	574
B_{gi}, formation volume factor, RB/MSCF	0.6822
c_{ti}, total compressibility, psi^{-1}	0.0001516
h, formation thickness, ft	36
L, horizontal well length, ft	1500
p_i, initial pressure, psi	4,700
k, permeability, md	8
f, porosity, fraction	0.09
r_w, wellbore radius, ft	0.3
S_g, gas saturation, fraction	0.5538
T, reservoir temperature, °R	640
m, viscosity, cp	0.022391
z_i, initial gas compressibility factor	0.99307

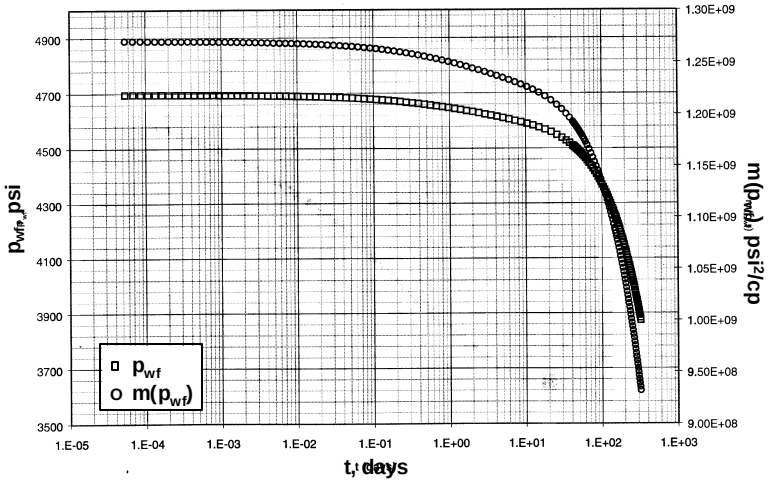

Fig. 5 – Pressure and pseudopressures vs. time for the example application of decline type-curve analysis.[4]

To start the analysis, original gas in place, G, should be estimated first. The estimation of G requires an iterative procedure as discussed above. For this example, an initial guess may be considered between the lower and upper bounds for G estimated as 19.03×10^6 and 23.25×10^6 MSCF, respectively. Using the initial guess for G and the material balance equation given by Eq. 20, the average pressures and, then, the productivity index, $q/\left[m(\bar{p})-m(p_{wf})\right]$, may be computed as a function of pseudoequivalent time, t_a. This procedure is repeated until a reasonably constant productivity index is obtained during the boundary-dominated flow. Figure 6 shows the iteration process and indicates that $G=21.14\times10^6$ MSCF is a good estimate for the GIP.

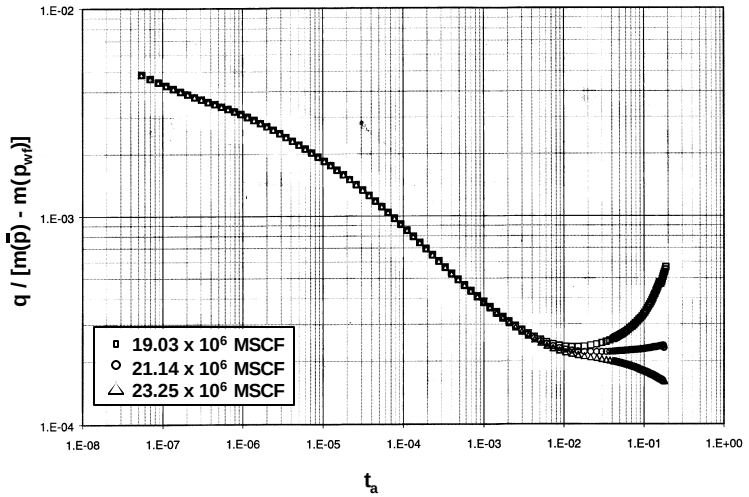

Fig. 6 – Iterative estimation of the original gas in place for the example application.[4]

Once a good estimate of G is obtained, then the corresponding productivity index may be analyzed by type-curve matching. Figure 7 shows the match of the productivity index data with the appropriate horizontal well type-curve.[4,5] The following match points may be chosen for the example analysis:

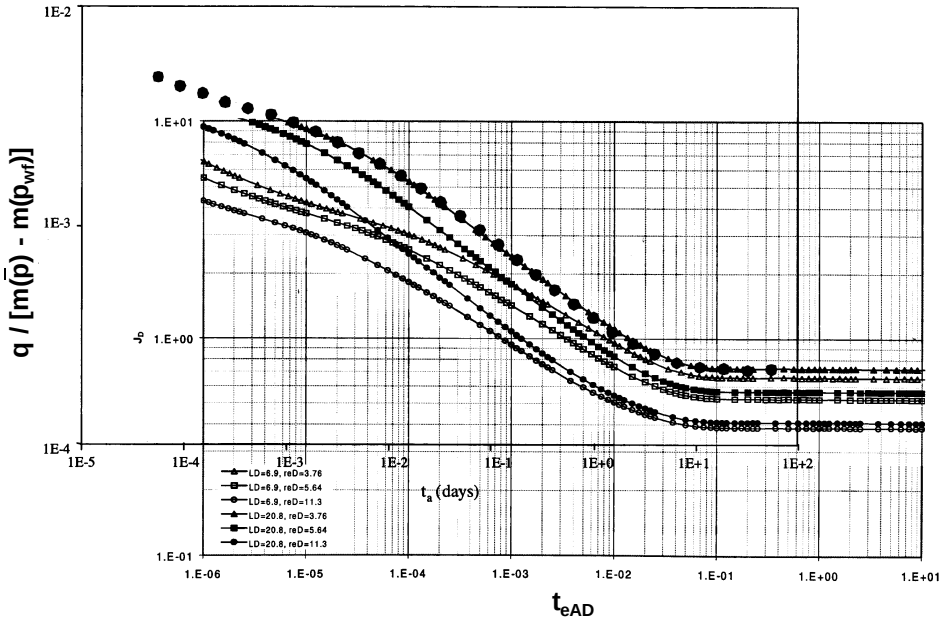

Fig. 7 – Decline type-curve analysis for the example application.[4]

$$[t]_{M.P.} = 1.5,$$

$$[t_{eAD}]_{M.P.} = 1\times 10^{-2},$$

$$\left\{\frac{q_g}{[m(\bar{p}) - m(p_{wf})]}\right\}_{M.P.} = 3\times 10^{-4},$$

and

$$[J_D]_{M.P.} = 0.98.$$

By using Eqs. 36 and 37, we estimate the following values of k and G, respectively:

$$k = 7.74\ mD,$$

and

$$G = 20.34 \times 10^6 \ MSCF \ .$$

It can be seen that the estimates of G from the iteration process and type-curve matching are in good agreement.

Although the theory developed above and the application example indicate that production data may be used to estimate the properties of reservoirs similar to pressure transient analysis, we have found that the application would normally suffer from the quality of the production data especially during transient flow periods.[3-5] As in most stripper gas wells, if the production data is based on daily gas measurements at the collector or separator, the quality of the data does not meed the requiremets of decline-type-curve analysis. Specifically for this project, we tested the production data provided by Marjo Operating Company and concluded that the quality of the data was insufficient for the application of decline-type-curve analysis. Therefore, for the stripper gas wells, the estimation of the reservoir properties need to be by conventional techniques (cores, logs, and pressure-transient tests) unless measures are taken to estimate gas production with accuracy close to that for pressure-transient measurements.

Task 1.2 – Development of Production Optimization Algorithm:

The main task of this project was to develop an optimization method for plunger lift operation based on reservoir performance. The idea behind this optimization method is to find the longest production period for which the liquid accumulated in the wellbore can be lifted by the pressure that builds up during the following shut-in period. To explain the production optimization algorithm, the following analysis of the plunger-lift operation may be helpful:

1. During shut-in, the plunger sits at the bottom of the wellbore with a liquid column from the previous production period above it (see Fig. 1). When the well is open to production, because of the pressure differential between the casing (which is equal to the sandface pressure of the reservoir) and the tubing-head pressure (which is mainly controlled by the line pressure), the plunger starts moving up and the liquid above the plunger is produced with the gas at the surface. If the pressure differential is sufficient to overcome the forces acting upon the fluids in the tubing, then the plunger may reach the surface, and all the liquid from the previous production period may be removed from the wellbore. Production of gas may continue after the plunger reaches the surface.

2. During the production period, while the liquid from the previous production period is removed by the lift of the plunger, new liquid in the produced gas starts accumulating in the tubing below the plunger. The height of the liquid column depends on the liquid content of the gas and the duration of the production period. During production, the sandface pressure (casing pressure)

drops because of the withdrawal of fluids from the reservoir and the tubing pressure increases because of the accumulation of the liquid.

3. Production stops either because the differential pressure between the sandface and the tubing becomes zero or the well is shut in (for successful plunger lift design, the pressure differential should not drop to zero before the plunger reaches the surface). At this time, the plunger starts descending in the tubing (the one-way valve on the plunger allows the liquid to rise above the plunger) until it rests at the bottom with the liquid column above it.

4. During the shut-in period, reservoir flow builds up the pressure at the sandface (casing) to a level sufficient to push the plunger and the liquid column above it to the surface.

The objective of the optimization is to determine the optimum production and buildup times to maximize the cumulative production. To maximize production, it is desirable to make the production periods longer and buildup periods shorter. Longer production periods, however, require longer shut-in times to build the pressure up to the level required to lift the fluids to the surface. Therefore, an optimum needs to be found for the production and shut-in times so that the cumulative production for a given sequence of production and shut-in periods is maximized. Because the optimization of the production time requires the knowledge on the buildup performance, which is chronologically later in the plunger-lift sequence, real-time measurements of tubing and casing pressures cannot be used in this optimization problem. We used analytical models of the wellbore and reservoir to simulate the production and buildup performances and developed an iterative algorithm to determine the optimum production and buildup times.

Below, we summarize the models used to simulate the wellbore hydraulics during plunger lift and reservoir performance during the production and shut-in periods. Next, we discuss the iterative algorithm to couple the wellbore hydraulics and reservoir performance to determine the optimum production and shut-in periods.

Wellbore Hydraulics for Plunger Lift: For the purposes of this project, we needed to know the change in the bottomhole pressure during production as a function of time. Specifically, we needed a model to simulate the effect of wellbore hydraulics on the bottomhole pressure during production. The results presented in Refs. 7 – 9 were helpful to simulate the wellbore hydraulics during plunger lift.

Foss and Gaul[7] presented the following pressure-balance equation when the plunger is rising in the tubing with its liquid load:

Casing pressure + pressure due to the weight of gas column
– gas frictional pressure drop in the annulus =
gas frictional pressure drop in the tubing under the plunger
+ pressure due to the weight of the gas column under the plunger
+ plunger frictional pressure drop
+ pressure required to lift the plunger weight
+ pressure required to lift the liquid weight
+ liquid frictional pressure drop
+ gas frictional pressure drop above the plunger
+ pressure due to the weight of the gas column above the plunger
+ surface tubing back pressure
+ pressure of the produced liquid under the plunger (38)

Although Eq. 38 indicates that the casing pressure changes from a maximum when the plunger starts rising from the bottom of the tubing to a minimum when the liquid slug and the plunger reaches the surface,[7-9] in our model we assume that the production period is characterized by a constant bottomhole pressure. From Eq. 38, it can be shown that the average casing pressure, $p_{c,avg}$, during production is given by[7]

$$p_{c,avg} = \left(1 + \frac{A_t}{2A_a}\right)\left(1 + \frac{D}{K}\right)p_L \qquad (39)$$

where

A_t: tubing cross-sectional area, ft^2,

A_a: annulus cross-sectional area, ft^2,

D: tubing depth, ft,

K: gas friction term, ft,

$$p_L = p_p + p_t + (p_{1h} + p_{1f})V_l + 14.7 \text{, psi,} \qquad (40)$$

p_p: pressure to lift the plunger weight, psi,

p_t: flowline pressure, psi,

p_{1h}: pressure to lift 1 bbl of fluid in the tubing, psi,

p_{1f}: frictional pressure loss per barrel of liquid, psi,

and

V_l : liquid volume above the plunger, bbl.

In Eq. 40, p_{1h} and p_{1f} may be computed from the following equations, respectively:[7,9]

$$p_{1h} = 0.433 g_l \left(\frac{5.615}{A_t} \right), \tag{41}$$

and

$$p_{1f} = \frac{0.433 g_l \, f_l \, v_p^2}{2(d_t/12)(32.2)(144)} \left(\frac{5.615}{A_t} \right), \tag{42}$$

where

g_l : specific gravity of the liquid

f_l : liquid friction factor in the tubing

v_p : average plunger velocity during production, ft/s,

and

d_t : tubing diameter, in.

The gas friction term, K, in Eq. 39 may be computed from the following equation[8,9]

$$\frac{1}{K} = \frac{f_g \, v_p^2 \, g_g}{2(d_t/12)(32.2)(T+460)(10.732)(144)Z}, \tag{43}$$

where

f_g : gas friction factor in the tubing,

g_g : specific gravity of gas,

T : average temperature in the tubing, °F,

and

Z : gas compressibility factor.

In this project, to compute the friction factor, f, for turbulent flow ($N_{Re} > 2300$), we used the Colebrook[10] correlation given by

$$\frac{1}{\sqrt{f}} = 1.74 - 2log\left(\frac{2e}{d_t} + \frac{18.7}{N_{Re}\sqrt{f}}\right), \tag{44}$$

where e is the surface roughness of the tubing and N_{Re} is the Reynolds number given by

$$N_{Re} = 4.71\times10^{-3}\frac{g\, v_p\, d_t}{(10.732)m(T+460)Z}. \tag{45}$$

For laminar flow ($N_{Re} \le 2300$), the friction factor was calculated from the following relation:

$$f = \frac{64}{N_{Re}}. \tag{46}$$

Another critical piece of information for plunger-lift optimization is the pressure when the plunger starts ascending from the bottom of the tubing (that is, the pressure at the beginning of the production period). This pressure should be sufficient to lift the plunger and its liquid load to the surface and is determined by the potential of the reservoir and the length of the pressure buildup period. If the liquid load from the previous production period is known, then the plunger-lift pressure model (Eq. 38) can provide the minimum pressure necessary to lift the plunger to the surface. Because this pressure is to be reached during the buildup period, a reservoir model can be used to determine the duration of the buildup period.

According to the Foss and Gaul[7] theory, when the plunger starts ascending in the tubing, the casing pressure is maximum and is given by[7,8]

$$p_{c,max} = 2p_{c,avg}\left(\frac{A_a + A_t}{2A_a + A_t}\right). \tag{47}$$

The casing pressure decreases as the plunger rises in the tubing and becomes a minimum when the plunger reaches the top of the tubing. The minimum casing pressure during plunger lift is given by[7,8]

$$p_{c,min} = 2p_{c,avg}\left(\frac{A_a}{2A_a + A_t}\right). \tag{48}$$

To determine the average and maximum casing pressures by using Eqs. 39 and 47, respectively, the liquid column height (and hence the production) must be known. Therefore, the plunger-lift optimization algorithm should relate Eqs. 39 and 47 to a

reservoir performance model. Below, we discuss the reservoir model used in the optimization algorithm.

Reservoir Performance Model for Plunger Lift: As noted above, to predict the reservoir performance during production, we assumed a constant flowing bottomhole pressure, $p_{wf,cp}$, equal to the average casing pressure, $p_{c,avg}$, in this project. This is for convenience for the optimization algorithm and may be justified based on the claim that the production does not change significantly during plunger lift production from tight gas wells even if the bottomhole pressure is changed.[9] Because our main interest is to determine the height of the liquid column, a reasonably good estimate of the cumulative production at the end of the production period must be sufficient for our purposes. (It should be noted that the same cumulative production would require different production times under constant rate and constant pressure production conditions. Our numerical experiments, however, indicated that the constant pressure production assumption would yield conservative estimates of the production times.)

Because the production (drawdown) period is followed by a shut-in (buildup) period in plunger lift operation, we first attempted to model a sequence of constant-pressure production followed by shut in. Although we were able to derive several approximate solutions, as discussed in our Third-Quarter Report,[11] the numerical evaluation of the solutions posed problems because of the discontinuity involved at the instant of shut-in. Similar problems, have also been reported in the literature.[12-15] In general, the solutions provided reasonably accurate buildup pressures for shut-in times much smaller than the producing time. In realistic plunger-lift operation conditions, however, producing times are much shorter than the buildup times. Therefore, we could not use the approximate analytical solutions for buildup pressures following production at a constant pressure. The alternative to a combined solution for the production and shut-in periods is to model each flow period separately.

Below, the production and buildup models used for reservoir pressure calculations are discussed. Following the standard procedures, gas-flow solutions in porous medium are presented in terms of pseudopressure, $m(p)$, defined by Eq. 19. Because the measurements in the field and the wellbore hydraulics model discussed above are in terms of pressure, the coupling of the wellbore and reservoir solutions is carried out in terms of pressure. This requires generating a table of pseudopressure versus pressure for the range of interest of the pressures. This table is then used to convert pressure to pseudopressure and vice versa by a table-look-up procedure with interpolation.

a) Production period:

Using Duhamel's equation,[6] the following relation between the pseudo pressure drop due to constant pressure production, $Dm_{wf,cp}$, and constant unit-rate production, $Dm_{wf,ur}$, may be written as follows:

$$Dm_{wf,cp}(t)=\int_0^t q(t)Dm'_{wf,ur}(t-t)dt\ , \tag{49}$$

where $q(t)$ is the gas production rate in MSCFD at the constant pressure $Dm_{wf,cp}$,

$$Dm'_{wf,ur}(t)=\frac{\partial Dm'_{wf,ur}}{\partial t}(t), \tag{50}$$

and

$$Dm_{wf}=m(p_i)-m(p_{wf}). \tag{51}$$

As discussed above, we assume that the constant flowing bottomhole pseudopressure, $Dm_{wf,cp}$, may be approximated by the pseudopressure corresponding to the average casing pressure, $p_{c,avg}$, given by Eq. 39 during production.

Evaluating the Laplace transform of Eq. 49, the following equations to calculate the gas production rate, $\overline{q}(s)$, and cumulative gas production, $\overline{Q}(s)$, in Laplace domain are obtained:

$$\overline{q}(s)=\frac{Dm_{wf,cp}}{s^2 D\overline{m}_{wf,ur}}\ , \tag{52}$$

and

$$\overline{Q}(s)=\frac{\overline{q}(s)}{s}. \tag{53}$$

where s is the Laplace transform parameter.

In Eq. 52, $D\overline{m}_{wf,ur}$ (the Laplace transform of the pseudopressure drop due to production at a constant unit rate) is given by[16]

$$D\overline{m}_{wf,ur}=\frac{1422T}{khs}\left\{\frac{I_0\left(\sqrt{s/h}\,r_w\right)K_1\left(\sqrt{s/h}\,r_e\right)+I_1\left(\sqrt{s/h}\,r_e\right)K_0\left(\sqrt{s/h}\,r_w\right)}{\sqrt{s/h}\,r_w\left[I_1\left(\sqrt{s/h}\,r_e\right)K_1\left(\sqrt{s/h}\,r_w\right)-I_1\left(\sqrt{s/h}\,r_w\right)K_1\left(\sqrt{s/h}\,r_e\right)\right]}+S\right\} \tag{54}$$

where I_0, K_0, I_1, and K_1 are the modified Bessel functions, r_w and r_e are the wellbore and reservoir radii (in ft), respectively, S is the skin factor, T is the average reservoir temperature in (°R), k is the permeability (in md), h is the formation thickness in (ft) and

$$h = \frac{2.637 \times 10^{-4} k}{f \overline{cm}}. \tag{55}$$

In Eq. 55, $\overline{cm}$ is the average compressibility viscosity product given by[6]

$$\overline{cm} = \frac{1}{t}\int_0^t c(\overline{p})m(\overline{p})dt\ , \tag{56}$$

where $\overline{p}$ is the average reservoir pressure. For our purposes, because producing times are usually short, we evaluated cm product at the initial pressure so that

$$\overline{cm} = c(p_i)m(p_i). \tag{57}$$

When the reservoir boundaries do not influence the well response: that is, when

$$t < 0.1\frac{\phi r_e^2}{h}, \tag{58}$$

the solution for $D\overline{m}_{wf,ur}$ given in Eq. 54 may be replaced by

$$D\overline{m}_{wf,ur} = \frac{1422T}{khs}\left[\frac{K_0\left(\sqrt{s/h}\ r_w\right)}{\sqrt{s/h}\ r_w K_1\left(\sqrt{s/h}\ r_w\right)} + S\right]. \tag{59}$$

For most practical cases we tested during this project, the production period satisfied the time condition given by Eq. 58 and we could use Eq. 59 instead of Eq. 54.

The solutions given in Eqs. 52, 53, 54, and 59 are in the Laplace transform domain and need to be numerically inverted into the real-time domain. We used the Stehfest's algorithm[17] discussed in Appendix A for numerical inversion. The solutions discussed above also assume that the pressure is uniform in the reservoir and equal to p_i at the beginning of the production period. In our application, each production period follows a period of shut-in, which may not be long enough to reach a stabilized pressure. We tested two approximations for this problem. The first approximation was to use the last shut-in pressure as the initial pressure (as in the modified isochronal testing of gas wells) and the second approach was to calculate an average pressure at the end of each production period, t_p, and use this average pressure as the initial pressure for the next production period. The average reservoir pressure was calculated from the following material balance equation:

$$m(\overline{p}) = m(p_i) - \frac{1422Th\,Q(t_p)}{24\phi h r_e^2}, \tag{60}$$

where p_i corresponded to the initial pressure for the first flow period and to the average pressure at the end of the previous flow period for the consecutive flow periods. The latter of the two approaches provided better results because it approximately satisfied the material balance while the first approach was completely arbitrary (although it is used in modified isochronal testing of gas wells.)

One final remark on the solution given in Eq. 54 is about the shape of the reservoir. In this project, we assumed that a cylinder could approximate the reservoir shape because for the practical cases we used to test the algorithm, the production periods were not long enough to feel the effect of the reservoir boundaries. This approximation, however, is not a limitation for the general application because analytical solutions for many different reservoir geometries are available in the literature and may be easily used instead of Eq. 54.

b) Buildup period

Although the analytical solution for pressure buildup following constant rate production is well known, changing flow rate complicates the solutions. The problem of varying flow rate prior to shut-in has been addressed in the pressure-transient analysis literature[6,18-21] and several practical approaches have been proposed. In this project, we followed the suggestion of Horner[18] and used the analytical solution for pressure buildup following a production period, t_p, at a constant rate equal to the last rate prior to shut-in, $q(t_p)$. In this approach, a modified producing time, $\tilde{t}_p$, calculated by

$$\tilde{t}_p = \frac{Q(t_p)}{q(t_p)}, \tag{61}$$

replaced the actual producing time, t_p. In Eq. 61, $Q(t_p)$ denotes the cumulative production during the flow period. We chose this approach because it is relatively simple and it satisfies the material balance. (It should, however, be noted that the justification for the use of this and the other approaches presented in the pressure-transient analysis literature is based on the existence of a longer straight-line on a Horner plot; not on the accuracy of the estimated pressures. In addition, producing times much longer than the buildup time are required.[6,18-21] Because our interest in this project is in the magnitude of the pressures, we used the above approach with reservations.)

With the above modification of the producing time, in this project we used the solution for pressure buildup following constant-rate production in Laplace domain presented by Correa and Ramey:[22].

$$D\overline{m}_{ws} = \frac{D\overline{m}_{wf,ur}}{s\left(1+24Cs^2 D\overline{m}_{wf,ur}\right)}\left(1-e^{-s\tilde{t}_p}\right), \tag{62}$$

where C is the wellbore storage coefficient defined by Eq. B-10 in Appendix B.

Because the solution given in Eq. 62 is in the Laplace transform domain, the results are numerically inverted into the real-time domain by using the Stehfest's algorithm[17] (Appendix A). However, because of the discontinuity involved at time $t = \tilde{t}_p$, the numerical inversion of the solution given by Eq. 62 poses problems. We used the approached suggested by Chen and Raghavan[23] to calculate the pressure buildup responses. A summary of the calculation procedure is presented in Appendix C.

Optimization Algorithm: The following ideas are used in the construction of the optimization algorithm: During production, the bottomhole flowing pressure, p_{wf}, is between the minimum and maximum casing pressures; that is,

$$p_{c,min} \le p_{wf} \le p_{c,max}. \tag{63}$$

We assume that p_{wf} may be represented by the average casing pressure, $p_{c,avg}$, given by Eq. 39:

$$p_{wf} \approx p_{c,avg}. \tag{64}$$

Similarly, at the end of the buildup period, we require that the buildup pressure, p_{ws}, at the sandface must be equal to or larger than the maximum casing pressure, $p_{c,max}$, given by Eq. 47; that is,

$$p_{ws} \ge p_{c,max}. \tag{65}$$

We note, however, that both $p_{c,avg}$ and $p_{c,max}$ are functions of the liquid volume accumulated in the tubing during the production period. The liquid accumulation, on the other hand, is a function of the Gas Liquid Ratio (*GLR*) and cumulative production, which, itself, is a function of the production period, t_p, and constant production pressure, $p_{wf} \approx p_{c,avg}$. Therefore, the calculations of $p_{c,avg}$ and $p_{c,max}$ require the knowledge of $p_{c,avg}$. This imposes the use of an iterative solution procedure.

There are several iterative loops in the calculations. We first consider the calculation of the cumulative production, $Q(t_p)$ for a given value of t_p (we will have to iterate on t_p later). To begin the calculations, we make an initial guess for $p_{c,avg}$ and use $p_{wf} \approx p_{c,avg}$ in Eqs. 52 and 53 to calculate $q(t_p)$ and $Q(t_p)$. Using the calculated cumulative production, $Q(t_p)$, and the known *GLR*, we calculate the liquid volume accumulated in the tubing by the following expression:

$$V_l = \frac{1000 \times Q}{5.615 \times GLR}(bbl). \tag{66}$$

Knowing the liquid volume in the tubing, V_l, we calculate $p_{c,avg}$ from Eq. 39 and compare with the assumed value in the beginning of the calculations. If the calculated and assumed pressures are within 1%, we accept the calculated $p_{c,avg}$ as the correct pressure and the corresponding $q(t_p)$ and $Q(t_p)$ values as the correct rate and cumulative production, respectively. If the difference between the calculated and assumed pressures is higher than 1%, then we use the calculated $p_{c,avg}$ as the new guess and repeat the process.

Once the cumulative production is known, the maximum bottomhole pressure, $p_{c,max}$, at the end of the production period can be calculated from Eqs. 39 and 47. The objective of the buildup period is, then, to keep the well shut in long enough for the reservoir pressure at the sandface, p_{ws}, to become equal to or higher than the bottomhole pressure, $p_{c,max}$. Because we approximate the buildup pressures by assuming that the production period was at a constant rate equal to $q(t_p)$ (the last rate prior to shut-in), we calculate an equivalent producing time, $\tilde{t}_p$, by using Eq. 61 and the buildup pressures from Eq. 62. In this process, we start with a shut-in time of $Dt = t - \tilde{t}_p = 0$ (no buildup period) and increase it until the calculated buildup pressure, p_{ws}, becomes equal to or higher than $p_{c,max}$.

We have found that the criterion used to stop the buildup calculations strongly influences the optimization results. After testing several options, we decided to use a logarithmic increment scheme for the shut-in pressures. We start, for example, with an initial increment of 0.01 hr and use this increment until the shut-in time, Dt, becomes 1 hr (of course, if p_{ws} becomes equal to or higher than $p_{c,max}$ before $Dt = 1$ hr, we stop calculations). When Dt, becomes 1 hr, we increase the time increment to 0.1 hr and continue with this increment until $Dt = 10$ hr, at which time we increase the time increment to 1. Although other schemes may be possible, we have found that this scheme to calculate the buildup pressures yielded a stable optimization algorithm without making excessive number of calculations (because of too small time increments) or yielding too large buildup times (because of too large time increments).

The calculation procedures discussed above were considered as independent of each other. In the optimization algorithm, we calculate the optimum production and shut-in times to maximize the cumulative production which makes these two times interrelated. This, however, requires another iterative solution. We start with a producing time of $t_p = 0.1$ hr and calculate Q and $p_{c,max}$ as described above. We start buildup with these values and $Dt = 0$ as discussed above and calculate the buildup pressure, p_{ws}.

If $p_{ws} > p_{c,max}$, this indicates that the reservoir has enough energy to lift the produced liquids so we can go back and increase the producing time, t_p, by using the same logarithmic increment scheme as described above for the buildup calculations. If $p_{ws} < p_{c,max}$, then the buildup pressure is not sufficient to lift the fluids in the tubing so we go back and increase the buildup time for the same producing time value. If $p_{ws} \approx p_{avg}$ without $p_{ws} > p_{c,max}$, where p_{avg} is calculated from Eq. 60 at the end of the previous producing period, this indicates that t_p is too large and the available reservoir energy will not be sufficient to lift the fluids to accumulate for this producing time.

The above iterative scheme normally yields more than one pair of t_p and Dt values because for longer shut-in times, longer producing times may be possible. However, if the objective is to maximize the cumulative production in the long run (for a sequence of production and buildup periods) instead of a single production period, having more flow periods (shorter buildup periods) in a fixed period should be desirable. To find the optimum between longer producing and shorter buildup periods, we assumed in the above optimization scheme that the new pair of the t_p and Dt values are acceptable if they provide at least 5% increase in the cumulative production compared to the previous times.

The optimization algorithm described above was translated into a computational code written in C++ language. The code included a control program to send the opening and closing signals to the wellhead at the times determined by the optimization algorithm. A listing of the computational code along with an example input data file is provided in Appendix D. A fully executable electronic copy is also included in the attachment. The computational code does not require any modification for individual applications. However, the data file needs to be modified for the properties of the individual well and reservoir. Table 2 shows an example of the data set required to run the optimization algorithm.

Table 2 – Example input data set (Marjo well).

Tubing ID = 1.995 in	c_f = 0.000002 psi^{-1}
Tubing OD = 2.375 in	T_{avg} = 100 °F
Casing ID = 4.09 in	γ_{liquid} = 0.834
Tubing Depth = 6365 ft	γ_{gas} = 0.720
Tubing ε_D = 0.00003	μ_{liquid} = 0.82
d_w = 6 in	GLR = 6365 scf/bbl
A = 160 Acres	C = 0.0000047 bbl/psi
h_{total} = 77 ft	S = 2.5
p_{avg} = 270 psi	$w_{plunger}$ = 8 lb
p_{line} = 29.9 psi	$v_{plunger}$ = 15.4 ft/s

Activity 2 - Field-Testing and Validation of the Proposed Method:

Task 2.1 – Building the Electronic Box:

To implement the above-described optimization method, manufacturing an electronic control system to communicate with the wellhead was required. The plunger lift control program is designed to run from a laptop computer with a National Instruments DAQCard-6024E data acquisition board installed. The board is connected with a ribbon cable to a National Instruments SCB-68 connector block. The connector block provides an interface for connections to pressure transducers and the well control box (Fig. 8).

Fig. 8 – The electronic control system for plunger-lift optimization.

The control program uses information provided by the operator to calculate the optimal producing and shut-in times for a plunger lift system. Two threads of execution exist in the program. The first thread measures pressure at a given time interval, currently set at one second. The second thread actually calculates the optimal times and operates the well.

To set up the system, the pressure transducers (Setra) are connected to the wellhead. One transducer is positioned to measure the casing pressure and the other to measure the tubing pressure. Standard pipe fittings and procedures are used in the connections.

The Setra transducers are powered by a 24VDC power supply located in the well control box. The red and black lines are power supply lines, while the green and white lines provide the return voltage to indicate pressure (Figs. 9.A – 9.C). The bare conductor is connected to ground. To minimize soldering, connector blocks are provided, one for

each transducer. The connector blocks act as junction between the pressure transducer, the power supply, and the data acquisition connector block.

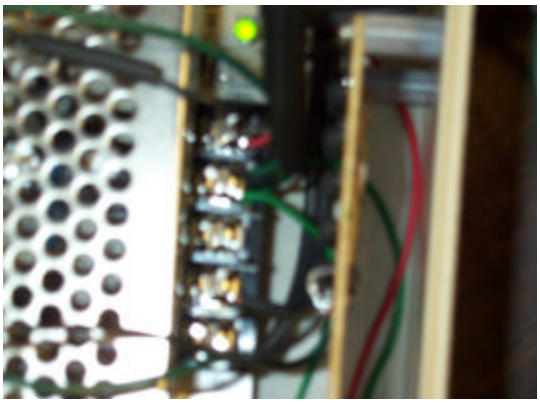

Fig. 9.A – Setra transducer connections.

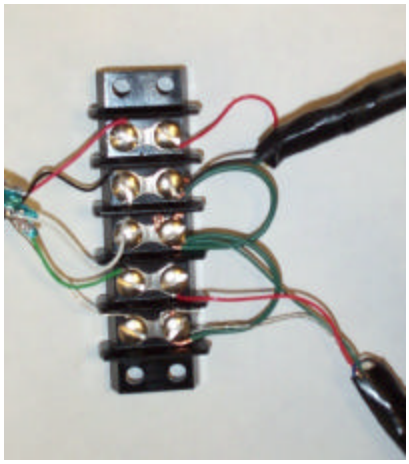

Fig. 9.B – Setra transducer connections.

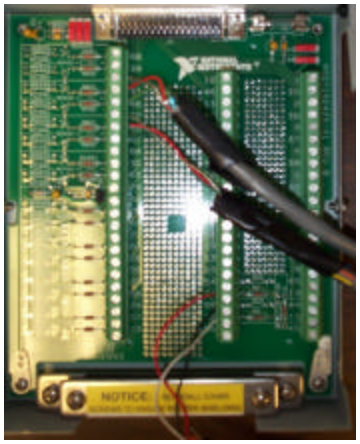

Fig. 9.C – Setra transducer connections.

The solenoid, used in the test wells provided by the Marjo Operating Company, is a 6-volt, three-wire, pulsed operating device. The middle conductor acts as the ground. To operate the solenoid, a 6-volt pulse is applied to the red conductor to open the valve. Applying a voltage pulse, controlled by a circuit, to the white conductor causes the solenoid to close (Fig. 10).

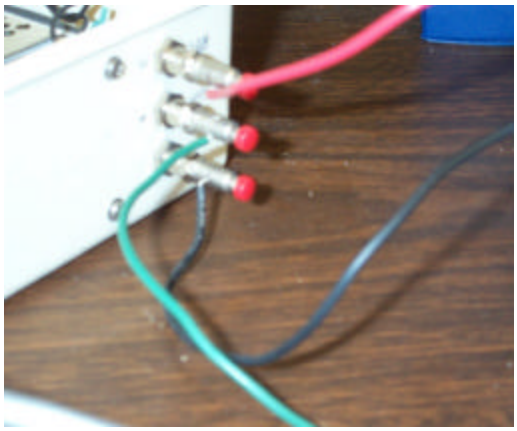

Fig. 10 – Connections between the solenoid and the control circuit.

Once the wiring connections have been made, the data acquisition board is plugged into the connector block with the ribbon cable. The computer and the power supply are also plugged. Because any power fluctuations adversely affect the program and ruin any test in progress, the computer is also plugged into a battery backup.

The computer includes the input data files, the timer program, and the plunger-lift optimization program. The operator at the well site can modify the data files to change or update the input parameters for the optimization algorithm. Starting the computer and the plunger lift program, starts the plunger lift production.

Task 2.2 – Field-Testing of the Method:

The optimization algorithm and the electronic box have been tested by using the field data first and then by applying on a well operated with plunger lift. Testing comprised of three stages: i) testing the algorithm, ii) testing the electronic control system, and iii) field implementation.

i) Testing the algorithm: At the first stage, the algorithm was tested by using the field data shown in Table 2 provided by the Marjo Operating Company. Figure 11 shows the sensitivity of cumulative gas production to reservoir parameters. As indicated by the figure, cumulative gas

production is very sensitive to the reservoir parameters supporting the initial motivation of this project to condition the optimization of plunger lift to the reservoir performance.

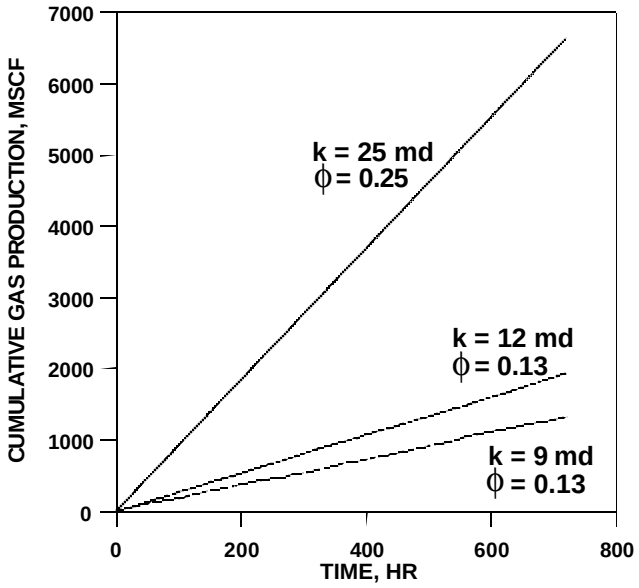

Fig. 11 – Sensitivity of cumulative gas production to reservoir properties.

For the well examined in this test, Marjo Operating Company had provided the average timer clock parameters as 0.4 hr of production and 4.5 hr of shut-in. We tested the corresponding cumulative productions resulting from the timer-clock operation and the optimization algorithm. For both cases, we used the analytical model developed in this project (in one case we forced the production and shut-in times to the values provided by Marjo Operating Company and in the other case, we allowed the optimization algorithm to choose the production and shut-in times). Figure 12 shows the comparison of the cumulative productions for the two cases for a period of one month. The optimization algorithm is shown to double the cumulative production compared with the timer-clock operation. This result confirms the original motivation of this project and indicates that the optimization algorithm has great potential to significantly increase the production from stripper gas wells by improving liquid lifting.

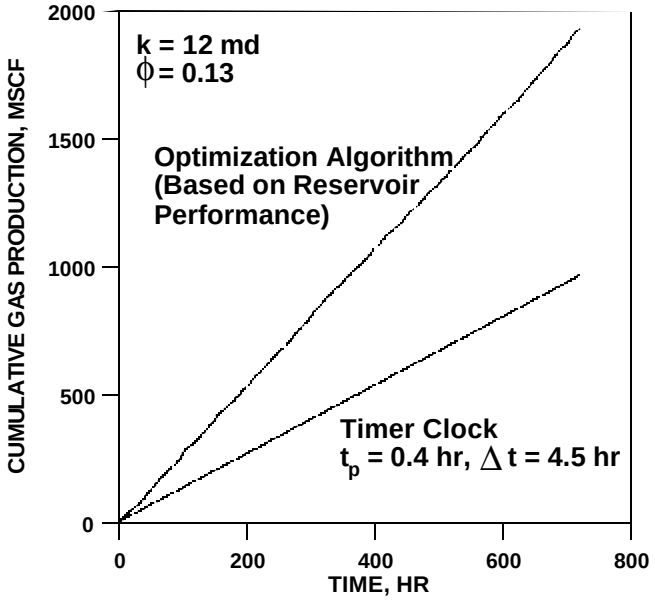

Fig. 12 – Comparison of timer-clock and optimization algorithm performances.

ii) At the second stage, the electronic control system was tested with the solenoid provided by Marjo Operating Company (the solenoid was chosen to be the same kind that is used in the well to be tested at the third stage.) The control system successfully communicated with the solenoid to pass the opening and closing signals in all tests. At the end of these tests, the electronic control system was approved for field-testing.

iii) At the third stage, the electronic control system and the optimization algorithm were implemented in the field at a plunger-lift operated well provided by Marjo Operating Company. Three tests were conducted on three different days. The tests were designed to be run with three different data sets to check the sensitivity to the data and to last eight to twelve hours. The control system failed in all three tests by not closing the well.

The first test appeared to have failed because of a battery shortage and/or improper voltage regulation (as discussed above, the solenoid used the well controls required exactly 6-volt pulse to recognize the opening and closing comments). The failures in the next two tests were attributed to the defects caused by the battery shortage in the first test on the electronic systems. The system was shipped to Colorado School of Mines and checked for any malfunctioning. All components of the electronic control system and the algorithm were tested and found to be working properly.

For this particular field test, the optimization algorithm did not require dynamic input from the well. Using the fixed input parameters, the optimization algorithm determined the opening and closing times of the

well and passed this information to the wellhead by the electronic control system. Because the well was opened first but never closed, the optimization algorithm itself was excluded from the list of potential reasons for the failure. The problem appeared to be the communication between the control system and the solenoid under the field conditions. Because the only difference between the office and field tests was the power supply, the control system was sent to the field again with stricter operational regulations in terms of power supply and voltage. However, the winter conditions in the test-well site have not permitted new tests until the due date of the final report. Further tests are planned when the weather conditions permits.

Task 2.3 – Final Report:

Throughout the project, the results have been compiled to construct this final project to be presented to the Stripper Well Consortium. Together with the quarterly reports, this final report documents the development and the results of the project.

CONCLUSIONS

In this project, an algorithm to optimize the production and shut-in times of plunger lift production from stripper gas wells has been developed and tested. In addition, potential of using production data to estimate the reservoir properties required by the optimization algorithm has been investigated. The following conclusions are derived as a result of this project:

1. Based on computer simulations, the optimization algorithm developed in this study based on reservoir performance has the potential to improve plunger lift production from stripper gas wells by 100%.

2. In addition to increasing production in standard operations, the proposed optimization algorithm can be used to automatically adjust the production and shut-in times of the plunger lift when the line pressure changes or fluctuates.

3. The control system required to implement the optimization algorithm in the field can be build under $1000. Therefore, the system should be in the reach of the small producers of stripper gas wells.

4. The implementation of the optimization algorithm does not require any modification of the existing wellheads. This minimizes the cost of implementation and causes minimum distraction to the continuing production.

5. The problems encountered in the field testing of the algorithm appear to be related to inadequate power supply. Further testing is required after correcting the unfavorable operating conditions.

6. Because of the demonstrated high potential of the proposed optimization approach, further research and testing is needed.

7. To commercialize the proposed method, continuous and reliable power supply must be integrated into the design of the electronic control system.

8. The use of decline-type-curve analysis techniques, such as the one developed in this project, has potential to estimate the reservoir properties from production history but accurate measurements of gas rates are required. Current rate measurement tools and practices on stripper gas wells do not usually meet the accuracy requirements for the success of the decline-type-curve analysis.

REFERENCES

1. Palacio, J. C. and Blasingame, T. A.: "Decline-Curve Analysis Using Type Curves: Analysis of Gas Well production Data," paper SPE 25909 presented at the 1993 Rocky Mountain Regional Meeting/Low Permeability Reservoirs Symposium and Exhibition, Denver, CO, April 26-28, 1993.

2. Agarwal, R., Gardner, D. C., Kleinsteiber, S. W., and Fussell, D. D.: "Analyzing Well Production Data Using Combined-Type-Curve and Decline-Curve Analysis Concepts," SPE Reservoir Evaluation and Engineering (October 1999) **2** (5), 478-486.

3. Chu, W. C., Cox, S. A., Gilbert, J. V., and McCarthy, C. J.: "Field Application of Production Analysis," paper SPE 62881 presented at the 2000 SPE Annual Technical Conference and Exhibition, Dallas, Texas, Oct. 1-4, 2000.

4. Araya, A.: Decline-Type-Curve Analysis of Horizontal Well Production Data, MSc. Thesis, Colorado School of Mines, Golden, CO (2002).

5. Araya, A. and Ozkan, E.: "An Account of Decline-Type-Curve Analysis of Vertical, Fractured, and Horizontal Well Production Data," paper SPE 77690 presented at the SPE 2002 Annual Technical Conference and Exhibition, San Antonio, TX, Sept. 29 – Oct. 2, 2002.

6. Raghavan, R.: *Well Test Analysis*, Prentice Hall, Englewood Cliffs (1993) 275,116,126.

7. Foss, D. L. and Gaul, R. B.: "Plunger-lift Performance Criteria with Operating Experience – Ventura Avenue Field," *Drilling and Production Practices*, API, 1965, 33-35.

8. Lea, J. F.: "Dynamic Analysis of Plunger Lift Operations," paper SPE 10253 presented at the SPE 1981 Annual Technical Conference and Exhibition, San Antonio, TX, Oct. 5 – 7, 1981.

9. Mower, L. N., Lea, J. F., Beauregard, E., and Ferguson, P. L.: "Defining the Characteristics and Performance of Gas-Lift Plungers," paper SPE 14344 presented at the SPE 1985 Annual Technical Conference and Exhibition, Las Vegas, NV, Sept. 22 – 25, 1985.

10. Colebrook, C. F.: "Turbulent Flow in Pipes with Particular Reference to the Transition Region Between the Smooth and Rough Pipe Laws," J. Inst. Civil Eng. (1939) 11, 133.

11. Ozkan, E.: "Optimization of Plunger-Lift Performance in Stripper Gas Wells," Quarterly Technical Report for the Period of November 15, 2001 – February 14, 2002, Stripper Well Consortium Project DE-FC26-00NT41025, The Pennsylvania State University, State College, PA, March 2002.

12. Ehlig-Economides, C. A.: "Pressure Buildup for Wells Produced at a Constant Pressure," paper SPE 7995 presented at the SPE 1979 California Regional Meeting, Ventura, CA, April 18-20, 1979.

13. Kutasov, I. M.: "Pressure Buildup for a Well in an Infinite Reservoir Produced at Constant Pressure," unsolicited manuscript SPE 12897, SPE, Richardson, TX (1984).

14. Kutasov, I. M.: "Application of the Horner Method for a Well Produced at Constant Pressure," unsolicited manuscript SPE 15143, SPE, Richardson, TX (1985).

15. Camacho-V., R., Rodriguez, F., Galindo-N., A., and Prats, M.: "Optimum Position for Wells Producing at Constant Wellbore Pressure," paper SPE 28715 presented at the SPE International Petroleum Conference and Exhibition of Mexico, Veracruz, Mexico, Oct. 10-13, 1994.

16. van Everdingen, A. F. and Hurst, W.: "The Application of the LaPlace Transformation to Flow Problems in Reservoirs," *Trans*. AIME 198 (1953) 171-176.

17. Stehfest, H.: "Numerical Inversion of Laplace Transforms," Communications ACM 13 No. 1 (1970) 47-49.

18. Horner, D. R.: "Pressure Build-Up in Wells," Proc. Third World Pet. Cong., The Hague (1951), Sec. II, 503-523.

19. Odeh, A. S. and Selig, F.: "Pressure Build-Up Analysis, Variable-Rate Case," J. Pet. Tech. (Sept. 1971) 1155-1160; Trans. AIME, 251.

20. Uraiet, A. A. and Raghavan, R.: "Pressure Build-Up Analysis for a Well Produced at a Constant Bottom-Hole Pressure," paper SPE 7984 presented at the SPE California Regional Meeting, Ventura, CA, April 18-20, 1979.

21. Ehlig-Economides, C. A. and Ramey, H. J., Jr.: "Pressure Buildup for Wells Produced at a Constant Pressure," paper SPE 7985 presented at the SPE California Regional Meeting, Ventura, CA, April 18-20, 1979.

22. Correa, A. C. and Ramey, H. J. Jr.: "Combined Effects of Shut in and Production: Solution with a New Inner Boundary Condition," paper SPE 15579 presented at the SPE 1986 Annual Technical Conference and Exhibition, New Orleans, LA, Oct. 5-8, 1986.

23. Chen, C. C. and Raghavan, R.: "An Approach to Handle Discontinuities by the Stehfest Algorithm," SPEJ (Dec. 1996) 363.

24. van Everdingen, A. F. and Hurst, W.: "The Application of the LaPlace Transformation to Flow Problems in Reservoirs," *Trans*. AIME 198 (1953) 171-176.

25. Ozkan, E., and Raghavan, R.: "Some New Solutions to Solve Problems in Well Test Analysis: I – Computational Considerations and Applications", SPE Formation Evaluation (September 1991) 359-368.

26. Peaceman, D.: "Interpretation of Well-Block Pressures in Numerical Reservoir Simulations with Nonsquare Grid Blocks and Anisotropic Permeability," *Soc. Pet. Eng. J.* (June 1983) 531-543.

27. Ozkan, E., Raghavan, R., and Joshi, S.D.: "Horizontal Well Pressure Analysis," *SPE Formation Evaluation*, Dec. 1989, 567-575.

28. Lee, J. and Wattenberger, R. A.: *Gas Reservoir Engineering*, SPE Textbook Series, Vol. 5, Richardson, TX, 1996.

LIST OF ACRONYMS AND ABBREVIATIONS

A : area, ft^2, Acres
B : formation volume factor, ft^3/SCF
C : wellbore storage coefficient, bbl-cp/psi^2
c : compressibility, psi^{-1}
d : diameter, ft
f : friction factor
G : gas in place, MSCF
h : formation thickness, ft
I_0 : modified Bessel function of the first kind of order zero
I_1 : modified Bessel function of the first kind of order one
J : productivity index, bbl/psi
K : gas friction term, ft
K_0 : modified Bessel function of the second kind of order zero
K_1 : modified Bessel function of the second kind of order one
k : permeability, md
L : horizontal well length, ft, Laplace transform
m : pseudopressure, psi^2/cp
N_{Re} : Reynolds number
p : pressure, psi
Q : cumulative production, bbl
q : production rate, bbl/d
r : radius, ft
S : Skin factor
s : Laplace transform parameter
T : temperature, ^oR
t : time, d, hr
V : volume, ft^3
Z : gas compressibility factor

Greek letters:

g : specific gravity
D : difference, shut in
m : viscosity
h : transmissibility coeffcient
f : porosity
e : surface roughness

Subscripts and superscripts

$\bar{}$: average, Laplace transform of
$\sim$: modified

a : annulus, pseudoequivalent
avg : average
c : casing
cp : constant pressure
D : dimensionless, tubing depth
e : external, equivalent
i : initial
l : liquid
max : maximum
min : minimum
M.P. : match point
p : producing
w : wellbore
wf : flowing wellbore
ws : shut in
t : total, tubing
ur : unit rate
0 : datum
-1 : inverse

APPENDIX A

ANALYTICAL SOLUTIONS FOR VERTICAL AND HORIZONTAL WELLS

In this appendix we present the analytical solutions for vertical and horizontal wells in cylindrical homogeneous reservoirs.

A.1. Analytical Solutions for Unfractured Vertical Wells in Cylindrical Reservoirs

Dimensionless pressure for fluid flow in a in a cylindrical porous medium of dimensionless radius r_{eD} is given in the Laplace transform domain by[24]

$$\overline{p}_D(r_D,s)=\frac{K_1\left(\sqrt{s}r_{eD}\right)I_0\left(\sqrt{s}r_D\right)+I_1\left(\sqrt{s}r_{eD}\right)K_0\left(\sqrt{s}r_D\right)}{s^{3/2}\left[I_1\left(\sqrt{s}r_{eD}\right)K_1\left(\sqrt{s}\right)+K_1\left(\sqrt{s}r_{eD}\right)I_1\left(\sqrt{s}\right)\right]}. \qquad \text{(A-1)}$$

In Eq. A-1, the overbar symbol indicates the Laplace transform of the function (that is, $\overline{p}_D$, is the Laplace transform of p_D), I_0, K_0, I_1, and K_1 are the modified Bessel's functions, s is the Laplace transform parameter, and the dimensionless radial distance r_D is defined by

$$r_D=\frac{r}{r_w}. \qquad \text{(A-2)}$$

van Everdingen and Hurst[24] obtained the following analytical inversion of Eq. A-1 by using the inversion integral:

$$\begin{aligned} p_D(r_D,t_D)=&\left\{\frac{2}{r_{eD}^2-1}\left(t_D-\frac{r_D^2}{4}\right)-\frac{r_{eD}^2}{r_{eD}^2-1}\ln r_D\right.\\ &-\frac{1}{4\left(r_{eD}^2-1\right)^2}\left(3r_{eD}^4-4r_{eD}^4\ln r_{eD}-2r_{eD}^2-1\right)\\ &\left.+p\sum_{n=1}^{\infty}\frac{J_1^2(b_n r_{eD})\left[J_1(b_n)Y_0(b_n r_D)-Y_1(b_n)J_0(b_n r_D)\right]}{b_n\left[J_1^2(b_n r_{eD})-J_1^2(b_n)\right]}exp\left(-b_n^2 t_D\right)\right\} \end{aligned} \qquad \text{(A-3)}$$

In Eq. A-3, b_1, b_2, etc. are the roots of

$$Y_1(b_n)J_1(b_n r_{eD})-J_1(b_n)Y_1(b_n r_{eD})=0, \qquad \text{(A-4)}$$

and, J_0, Y_0, J_1, and Y_1 are the Bessel's functions.

The solution given in Eq. A-1 may also be inverted numerically by using the Stehfest algorithm.[17] This algorithm obtains an approximate inverse, $p_a(T)$, of the Laplace domain function, $\overline{p}(s)$, at time $t = T$ by

$$p_a(T) = \frac{ln\,2}{T}\sum_{i=1}^{N} V_i\,\overline{p}(s)_{s=i\frac{ln\,2}{T}}\,, \tag{A-5}$$

where

$$V_i = (-1)^{(N/2)+i}\sum_{k=(i+1)/2}^{min(i,N/2)} \frac{k^{N/2}(2k)!}{[(N/2)-k]!\,k!(k\text{-}1)!(i\text{-}k)!(2k\text{-}i)!}, \tag{A-6}$$

and N is an even integer. Theoretically, the accuracy of the inversion should increase as N increases but the accuracy becomes less for large N because of round-off errors. Therefor, the optimum value of N needs to be determined by trial and error ($6 \le N \le 18$ is a common range for transient flow problems).

In this project, we used Stehfest's algorithm to develop the decline type curves. To compute the dimensionless wellbore pressures, p_{wD}, we set $r_D = 1$ in Eq. A-1. Note that the dimensionless pressure solution given by Eq. A-1 corresponds to production at a constant rate. To obtain the dimensionless flow rates, q_D, as a result of constant pressure production, we used the constant rate production solution given by Eq. A-1 with the following Laplace transform property:

$$\overline{q}_D\,\overline{p}_{wD} = \frac{1}{s^2}\,, \tag{A-7}$$

A.2. Analytical Solutions for Horizontal Wells in Cylindrical Reservoirs

Ozkan and Raghavan[25] have shown that the wellbore pressures of horizontal wells in a closed cylindrical reservoir can be computed from the following expression:

$$\overline{p}_{wD}(|x_D| \le 1, s) = \frac{1}{2s^{3/2}}\left[\int_0^{\sqrt{s}(1+x_D)} K_0(x)dx + \int_0^{\sqrt{s}(1-x_D)} K_0(x)dx\right] + \overline{F}(x_D, z_D, z_{wD}, L_D) \tag{A-8}$$

where

$$\overline{F}(x_D, z_D, z_{wD}, L_D) = \frac{1}{s}\sum_{n=1}^{\infty}\frac{\cos npz_D \cos npz_{wD}}{e_n} \left[\int_0^{\sqrt{s+n^2p^2L_D^2}(1+x_D)} K_0(x)dx + \int_0^{\sqrt{s+n^2p^2L_D^2}(1-x_D)} K_0(x)dx\right] \quad \text{(A-9)}$$

In Eqs. A-8 and A-9, the dimensionless variables are defined based on the half-length of the horizontal well, $L_h/2$, as follows:

$$p_D = \frac{k\,h}{141.2qB\mu}(p_i - p), \quad \text{(A-10)}$$

$$t_D = \frac{2.637\times10^{-4}kt}{\phi c_t \mu (L_h/2)^2}, \quad \text{(A-11)}$$

$$L_D = \frac{L_h}{2h}\sqrt{\frac{k_z}{k}}, \quad \text{(A-12)}$$

$$z_D = \frac{z}{h}, \quad \text{(A-13)}$$

$$r_{wD} = \frac{r_{w,eq}}{h}, \quad \text{(A-14)}$$

and

$$x_D = \frac{2x}{L_h}\sqrt{\frac{k_x}{k}}\ . \quad \text{(A-15)}$$

In Eqs. 12.3.1 and 12.3.2, k represents the geometric average of the principal permeabilities, k_x, k_y, and k_z that are assumed to be in the directions of the coordinate axes; that is, $k = \sqrt[3]{k_x k_y k_z}$. In Eq. A-14, $r_{w,eq}$ denotes the equivalent wellbore radius for an anisotropic medium and may be obtained from[26]

$$r_{w,eq} = 0.5 r_w \left[\left(k_y / k_z \right)^{0.25} + \left(k_z / k_y \right)^{0.25} \right]. \qquad \text{(A-16)}$$

The dimensionless variable x_D in Eqs. A-8 and A-9 determines the point to calculate the pressure along the well. For long horizontal wells, $x_D = 0.732$ yields the approximate response of an infinite-conductivity horizontal well.[27]

APPENDIX B

DERIVATION OF THE COEFFICIENT OF GAS STORAGE IN THE WELLBORE

In this appendix we summarize the derivation of the annulus gas storage coefficient for the buildup period of a plunger lift operation. It has been shown that the mass balance in a wellbore yields[28]

$$q_{sf} = q - q_{wb}, \tag{B-1}$$

where q_{wb} is the flow rate due to production of the fluid stored in the wellbore and q_{sf} and q denote, respectively, the sandface and surface production rates. The wellbore flow rate is given by

$$1000\, q_{wb}\, r_{sc} = -(24)(5.615) V_{wb} \frac{d\overline{r}_{wb}}{dt}. \tag{B-2}$$

In Eq. B-2,

q_{wb} : gas flow rate due to storage, $MSCFD$,

t : time, hr,

$\overline{r}_{wb}$: average density of the fluid in the wellbore, lb_m / ft^3,

r_{sc} : density of the fluid at standard conditions, lb_m / ft^3, and

V_{wb} : volume of the fluid stored in the wellbore, bbl.

Using

$$\frac{d\overline{r}_{wb}}{dt} = \frac{d\overline{r}_{wb}}{d\overline{p}_{wb}} \frac{d\overline{p}_{wb}}{dt} = \overline{c}_{wb}\, \overline{r}_{wb} \frac{d\overline{p}_{wb}}{dt} \tag{B-3}$$

we can write Eq. B-3 as follows:

$$q_{wb} \frac{r_{sc}}{\overline{r}_{wb}} = -(24)(5.615 \times 10^{-3}) \overline{c}_{wb} V_{wb} \frac{d\overline{p}_{wb}}{dt} \tag{B-4}$$

For the shut-in period of plunger lift, the fluid is stored in the annulus between the casing and tubing and may be assumed as single-phase gas. Thus,

$$V_{wb} = (A_c - A_t) h, \tag{B-5}$$

where A_c and A_t are the cross-sectional areas of the casing and tubing, respectively, and h is the formation thickness. From the gas equation of state, we can write[28]

$$B_g = \frac{r_{sc}}{\overline{r}_{wb}} = \frac{\overline{Z}_{wb} \overline{T}_{wb}\, p_{sc}}{\overline{p}_{wb} T_{sc}} = 0.02827 \frac{\overline{Z}_{wb} \overline{T}_{wb}}{\overline{p}_{wb}}, \tag{B-6}$$

where we have used $p_{sc} = 14.7\,psi$ and $T_{sc} = 520^{\circ} R$. Substituting Eqs. B-5 and B-6 into Eq. B-4, we obtain

$$q_{wb} = -24\frac{\overline{p}_{wb}\overline{c}_{wb}(A_c - A_t)h}{5.035\overline{Z}_{wb}\overline{T}_{wb}}\frac{d\overline{p}_{wb}}{dt}. \tag{B-7}$$

Because

$$\frac{d\overline{p}_{wb}}{dt} = \frac{\overline{m}_{wb}\overline{Z}_{wb}}{2\overline{p}_{wb}}\frac{d\overline{m}_{wb}}{dt} \tag{B-8}$$

Eq. B-7 may be written as

$$q_{wb} = -24\frac{\overline{m}_{wb}\overline{c}_{wb}(A_c - A_t)h}{10.07\overline{T}_{wb}}\frac{d\overline{m}_{wb}}{dt}. \tag{B-9}$$

Thus, if we define a wellbore storage coefficient by

$$C = \frac{\overline{m}_{wb}\overline{c}_{wb}(A_c - A_t)h}{10.07\overline{T}_{wb}}, \tag{B-10}$$

then, we can write Eq. B-9 as follows:

$$q_{wb} = -24C\frac{d\overline{m}_{wb}}{dt}. \tag{B-11}$$

APPENDIX C

NUMERICAL INVERSION OF THE BUILDUP SOLUTION GIVEN BY EQ. 62

This appendix summarizes the numerical inversion procedure used to calculate the pressure buildup responses from Laplace domain solution given by Eq. 62 in the text. In Petroleum Engineering, the numerical inversion algorithm due to Stehfest[17] is commonly used to compute the real time values of the solutions in the Laplace transform domain. In Eq. 62, discontinuity of the production caused by the shut-in at $t = \tilde{t}_p$ poses difficulties in the numerical inversion.

The discontinuity at $t = \tilde{t}_p$ is represented by the $e^{-s\tilde{t}_p}$ terms in Eq. 62. Chen and Raghavan[23] have suggested that for the numerical inversion of a function in the form of $\bar{f}(s)\left(1-e^{-s\tilde{t}_p}\right)$ by using Stehfest's algorithm, the following formula should be useful:

$$L^{-1}\left\{\bar{f}(s)\left(1-e^{-s\tilde{t}_p}\right)\right\}_t = L^{-1}\left\{\bar{f}(s)\right\}_t - L^{-1}\left\{\bar{f}(s)\right\}_{t-\tilde{t}_p}. \qquad \text{(C-1)}$$

In Eq. C-1, L^{-1} denotes the inverse Laplace transformation. The first term in the right hand side of Eq. C-1 is inverted at time t and the second term is considered only when $t > \tilde{t}_p$ and is inverted at $t - \tilde{t}_p$.

Because our interest in this project is to compute the pressure buildup responses, we need to evaluate Eq. 62 for $t \geq \tilde{t}_p$. Then using Eq. C-1, the numerical inversion formula for Eq. 62 is written as follows:

$$L^{-1}\{D\overline{m}_{ws}\} = L^{-1}\left\{\frac{D\overline{m}_{wf,ur}}{s\left(1+24Cs^2D\overline{m}_{wf,ur}\right)}\right\}_t - L^{-1}\left\{\frac{D\overline{m}_{wf,ur}}{s\left(1+24Cs^2D\overline{m}_{wf,ur}\right)}\right\}_{t-\tilde{t}_p} \qquad \text{(C-2)}$$

In practice, Eq. C-2 is evaluated in two steps. In the first step, the first term in the right hand side of Eq. C-2 is inverted to the real-time domain at time t and the second term is inverted at time $t - \tilde{t}_p$ separately. In the second step, the difference between the numerical inversions of the first and second terms in the right hand side of Eq. C-2 is calculated as the value of Dm_{ws} in the real time domain.

APPENDIX D

COMPUTATIONAL CODE FOR THE OPTIMIZATION ALGORITHM

This appendix presents a listing of the computational code for the plunger lift optimization algorithm in C++ and an example input data file. An executable copy of the program is also provided in the attachment of this report.

A. Optimization Algorithm:

```
//
//  PLOP Version 1.0
//


/*
 * Includes:
 */
#include <conio.h>
#include <stdio.h>
#include <windows.h>

#include "nidaqex.h"
#include "PLOP.h"



/* Global Variable */
FILE *pressFP;    //File pointer for pressures


/* This function is a thread which reads and records voltage
 * from the transducer.  Modified from Microsoft and NI sample codes.
 */
DWORD WINAPI ReadPressFunc( LPVOID lpParam )
{

      /*
    * Local Variable Declarations:
    */
```

```
    i16 iStatus = 0;
    i16 iRetVal = 0;
    i16 iDevice = 1;
    i16 iChan = 1;
    i16 iGain = 1;
    f64 dVoltage = 0.0;
    i16 iIgnoreWarning = 0;

      int i;
      SYSTEMTIME locoTime;
      double dPressure1, dPressure2;

//    for (i=0;i<10;i++) {
      for (;;) {

            GetLocalTime(&locoTime);          // get current time

            iChan = 1;
        iStatus = AI_VRead(iDevice, iChan, iGain, &dVoltage);

            iRetVal = NIDAQErrorHandler(iStatus, "AI_VRead", iIgnoreWarning);

            dPressure1 = (dVoltage-0.11)*200.;          // convert to pressure


            iChan = 2;
            iStatus = AI_VRead(iDevice, iChan, iGain, &dVoltage);

            iRetVal = NIDAQErrorHandler(iStatus, "AI_VRead", iIgnoreWarning);

            dPressure2 = (dVoltage-0.11)*200.;          // convert to pressure

      // Build a string showing the date, time, and pressure measurements
            fprintf(pressFP, "%02d/%02d/%d  %02d:%02d:%02d  PT1 = %lf PT2 = %lf\n",
                  locoTime.wDay, locoTime.wMonth, locoTime.wYear,
                  locoTime.wHour, locoTime.wMinute, locoTime.wSecond,
                  dPressure1, dPressure2);
```

```
                Sleep(1000);  //Measurement approx. every second

        }

}

const long HTMS = 3600000;

void cal_production_data2(double &TP, double &DT){

return;
}

void OpenWell(double TP)
{
        printf("Opening the well for %lf hours.\n", TP);
        fprintf(pressFP,"Opening the well for %lf hours.\n", TP);
        /*
         * This function works by raising the voltage for 500 ms on DAC1
         */

    /*
     * Local Variable Declarations:
     */

    i16 iStatus = 0;
    i16 iRetVal = 0;
    i16 iDevice = 1;
    i16 iChan = 1;
     f64 dVoltage1 = 5.0;
     f64 dVoltage2 = 0.0;
    i16 iIgnoreWarning = 0;

        /* First output 5.0 volts to start pulse. */
    iStatus = AO_VWrite(iDevice, iChan, dVoltage1);
```

```
    iRetVal = NIDAQErrorHandler(iStatus, "AO_VWrite", iIgnoreWarning);

       Sleep(500);

    /* Then output 0.0 volts to end pulse. */
    iStatus = AO_VWrite(iDevice, iChan, dVoltage2);

    iRetVal = NIDAQErrorHandler(iStatus, "AO_VWrite", iIgnoreWarning);

}

void CloseWell(double DT)
{
       printf("Shut-in the well for %lf hours.\n", DT);
       fprintf(pressFP,"Shut-in the well for %lf hours.\n", DT);
       /*
        * This function works by raising the voltage for 500 ms on DAC0
        */

    /*
     * Local Variable Declarations:
     */

    i16 iStatus = 0;
    i16 iRetVal = 0;
    i16 iDevice = 1;
    i16 iChan = 0;
     f64 dVoltage1 = 5.0;
     f64 dVoltage2 = 0.0;
    i16 iIgnoreWarning = 0;

       /* First output 5.0 volts to start pulse. */
    iStatus = AO_VWrite(iDevice, iChan, dVoltage1);

    iRetVal = NIDAQErrorHandler(iStatus, "AO_VWrite", iIgnoreWarning);

       Sleep(500);
```

```
   /* Then output 0.0 volts to end pulse. */
   iStatus = AO_VWrite(iDevice, iChan, dVoltage2);

    iRetVal = NIDAQErrorHandler(iStatus, "AO_VWrite", iIgnoreWarning);

}

/* This function is a thread which operates the well based on values
 * calculated from cal_production_data
 */
DWORD WINAPI OperateWellFunc( LPVOID lpParam )
{

      double TP,DT;
      int count = 1;

      for(;;) {
                  printf("Run number %d.\n",count++);
                  //Calculates DP, TP and anything else
//                cal_production_data(TP, DT);
// What is the output????????????????????? Hours?? HTMS = 3600000
                  cal_production_data(TP, DT);
                  printf("TP= %lf hours, DT= %lf hours.\n",TP,DT);
                  //open well function
                  OpenWell(TP);
                  //Sleep (TP hrs)
                  Sleep(TP*HTMS);
                  //Close well function
                  CloseWell(DT);
                  //Sleep (DT hrs)
                  Sleep(DT*HTMS);
                  printf("\n");
      }
}

int main(int argc, char* argv[])
```

```
{

  DWORD dwThreadId1, dwThreadId2, dwThrdParam = 1;
  HANDLE hThread1, hThread2;
  char szMsg[80];

  printf("Hit any key to end program....\n");

    /* Open a file to put pressure into */
    pressFP = fopen("test1.txt","a");


  hThread1 = CreateThread(
     NULL,                    // no security attributes
     0,                       // use default stack size
     ReadPressFunc,           // thread function
     &dwThrdParam,            // argument to thread function
     0,                       // use default creation flags
     &dwThreadId1);           // returns the thread identifier

 // Check the return value for success.

 if (hThread1 == NULL)
 {
   wsprintf( szMsg, "CreateThread (ReadPressFunc) failed." );
   MessageBox( NULL, szMsg, "main", MB_OK );
     return 0;
 }

  hThread2 = CreateThread(
     NULL,                    // no security attributes
     0,                       // use default stack size
     OperateWellFunc,         // thread function
     &dwThrdParam,            // argument to thread function
     0,                       // use default creation flags
     &dwThreadId2);           // returns the thread identifier
```

```
      // Check the return value for success.

      if (hThread2 == NULL)
      {
        wsprintf( szMsg, "CreateThread (OperateWellFunc) failed." );
        MessageBox( NULL, szMsg, "main", MB_OK );
          return 0;
      }

        //Wait for a character to be pressed.
      _getch();

        CloseHandle( hThread1 );
        CloseHandle( hThread2 );

        /* Close file */
        fclose(pressFP);

        return (0);
    }

/* End of program */

    //
    //              FILE PLOP.h  10/31/2002
    //
    //  PROGRAM OPTIMIZES THE PRODUCTION AND BUILD-UP TIMES OF A
    //    PLUNGER-LIFT SYSTEM PRODUCING GAS IN THE PRESENCE OF A
    //    LIQUID COLUMN IN THE WELLBORE.
    //

    #include <iostream>
    #include <iomanip>
    #include <fstream>
    #include <cmath>
    #include <cstdlib>
    using namespace std;
```

```
const float HRS = 216000;

double FC(double, double);

void STEHFESTV(double [], int);

void PBUP(double &, int,double [],double, double, double, double,
          double, double, double, double, double, double, double, int);

double PU(double, double, double, double, double, double, int, double,
          double, double, double);

void QCPAV (double &, double &, double &, double &, double &, double &, int &,
            double &, int, double [], double, double, double, double, double,
            double, double, double, double, double, int, double, double, double,
            double, double, double, double, double, double, double, double,
            double, double, double, double, double, double [], double [], int);

void PROP(double &, double &, double, double, double, int);

void CPROD(double &, double &, int, double [], double, double, double, double, double, int);

void PCAVSUB(double &, double &, double, double, double, int, double, double, double,
             double, double, double, double, double, double, double, double, double,
             double, double, double);

double Q(double, double, double, double, double, int);

double BESSK1(double);

double BESSI1(double);

double BESSK0(double);

double BESSI0(double);

double ZFAC (double, double, double, int, int);
```

```
double GASVIS (double, double, double, int);

void dranchukcorr(double&, double, double);

void hallyarbcorr(double&, double, double);

void standingcorr(double&, double, double);

void gopalcorr(double&, double, double);

double FLAGR (double [], double [], double, int, int, int);

double YEST(double [],double [], int, int, double);

   double D, H,P, T, X;
     double TID, TOD, CID, WP, PTMIN, TIM E, QGCTOT, PSTART, PSPIN;
     int MCODE, ICOUNT, N, IERR, NPRES, IFR;
     double SGL, SGG, VPAV, TAV, GLR, VISL, EPSD, DPAVG, PI;
     double PIN, PERM, PHI, CF, CWS, SK, RW, RE, PMIN, PMAX, ETA, DMWS;
     double AT, AA, PP, BL, P1H, RHOL, REL, FL, WM, SUMP, CG, VISG, PCAV, TPCP;
   double PCAVN, PCMAXN, PCMAX, DPSPCN, QGCT, TT, CH1, CH2;
     double TPT, DTT, PCAVT, PCMAXT, PWS, PWST, QTP, QTPT, QTPP, PSPWSN, PWSN;
     double DQCN, QGC, PCAVO;
     double DPN, TN, TPN, DTN;
     double QGCN, QTPN ;


//*&^*^$&&#&%^%$^&&%$#%#^&%^#&$^*&^*%^&*&*^%&^&(&%*(&%(%#&$^%$#^&
//%$#^#%#$%^&$^&^&%^&*^&*%^*^%$#W%$^@@^@^$%^@^@%^%^$%^$%^%^@^%%%
%
//&*^%&)&)%%#(&%#(&#(&(&(%(%#&)#^(%@%^@(^$(*^%*%^$%^&^@*(^@(^$@))

void cal_production_data(double &TP, double &DT){

//
ifstream infil ("PLOdat3M.text", ios::in);
```

```
ofstream outfil ("PLOout1.text", ios::out);

const int arraySize1 = 51;
double V[arraySize1]={0.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
        1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.};
const int arraySize2 = 51;
double PRE[arraySize2]={0.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
        1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.};
double PSP[arraySize2]={0.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
        1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.};
double ZZ[arraySize2]={0.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
        1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.};
double VIS[arraySize2]={0.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
        1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
                1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.};



//    ***************************************************************
//    *          READ THE CONSTANT INPUT DATA (TABLE 1)          *
//    ***************************************************************
   infil >> TID >> TOD >> CID >> D >> WP >> PTMIN;
     infil>>SGL>>SGG>>VPAV>>TAV>>GLR>>VISL;
     infil>>EPSD;
```

```
    infil>>PIN>>PERM>>PHI>>CF>>CWS>>SK>>RW>>RE>>H;
    infil>>PMIN>>PMAX>>NPRES>>N;
//  ************************************************************
//  *       CALCULATE THE CONSTANT INPUT DATA (TABLE 2)        *
//  ************************************************************

    MCODE=1;
    ICOUNT=0;
    PI=4*atan(1);
    AT=PI*TID*TID/4/144;
    AA=PI*(CID*CID-TOD*TOD)/4/144;
    PP=WP/AT/144;
    BL=5.615/AT;
    P1H=0.433*SGL*BL;
    RHOL=0.433*SGL;
    REL=3.527*RHOL*VPAV*TID/VISL;
    FL=FC(REL,EPSD);
    WM=28.97*SGG;
//  ************************************************************
//  *    CALCULATE THE COEFFICIENTS OF THE STEHFEST ALGORITHM  *
//  ************************************************************
    STEHFESTV (V,N);
//  ************************************************************
//  *        INITIALIZE TIME AND CUMULATIVE PRODUCTION         *
//  ************************************************************
    TIME=0;
    QGCTOT=0;
//  ************************************************************
//  *          CONSTRUCT THE PSEUDOPRESSURE TABLE              *
//  ************************************************************
  DPAVG=(PMAX-PMIN)/(NPRES-1.);
  PSTART=PMIN;
    PRE[0]=0.;
    PSP[0]=0.;
    ZZ[0]=0.;
    VIS[0]=0.;
  SUMP=0.;
```

```
    for(int i=0; i < NPRES+1; ++i) {
            int ic=i+1;
            PRE[ic]=PSTART;
            ZZ[ic]=ZFAC(TAV,PSTART,SGG,MCODE,IERR);
            VIS[ic]=GASVIS(TAV,SGG,PSTART,IERR);
            if (ic==1)
                    SUMP=SUMP+(PRE[ic]/VIS[ic]/ZZ[ic])*PRE[ic];
            else
                    SUMP=SUMP+(PRE[ic-1]/VIS[ic-1]/ZZ[ic-1]+PRE[ic]/VIS[ic]/ZZ[ic])
                    *(PRE[ic]-PRE[ic-1]);

            PSP[ic]=SUMP;
            PSTART=PSTART+DPAVG;
    }

//  ****************************************************************
//  *CALCULATE THE PSEUDO INITIAL PRESSURE FOR THE FIRST         *
// *PRODUCTION CYCLE
              *
//  ****************************************************************
    PSPIN=FLAGR(PRE,PSP,PIN,1,NPRES,IERR);
//  ****************************************************************
//  *CALCULATE THE FIRST GUESS FOR THE CONSTANT WELLBORE PRESSURE  *
// *BY ASSUMING THAT P-INITIAL CORRESPONDS TO PC-MAX.           *
// ****************************************************************
    PCAV=PIN*(2*AA+AT)/2/(AA+AT);
// ****************************************************************
//  *CALCULATE TP AND DT TO OPTIMIZE THE PRODUCTION FOR A GIVEN    *
// *CYCLE. REPEAT CALCULATIONS UNTIL A STOPPING TIME                        *
                                                                *
//  ****************************************************************
//869048649086-860-86-3863850-809380383-    //while (TIME <= 720.){

// ****************************************************************
//  *PRODUCING TIME (DRAWDOWN AT A CONSTANT PRESSURE) CALCULATIONS *
                                                                *
//  ****************************************************************
```

```
                QGCT=1E-7;
                TT=0.1;
                TPCP=0.1;
                CH1=0.01;
    // ***************************************************************
    //    *CALCULATE THE CUMULATIVE PRODUCTION AND RATE (ASSUMING
*
    // *PRODUCTION AT CONSTANT PRESSURE PWF=PCAV) UNTIL PRESSURE     *
    // *DROPS BELOW PCMAX
*
    //    ***************************************************************

    start1:

                QCPAV(QGCN,QTPN,PCMAXN,PCAVN,CG,ETA,IFR,DPSPCN,N,V,
                        SK,RW,RE,PERM,PHI,H,VISG,TAV,PCAV,SGG,MCODE,WM,
                            TID,EPSD,TPCP,GLR,SGL,FL,VPAV,BL,PP,PTMIN,P1H,
                            AT,AA,D,PSPIN,PRE,PSP,NPRES);
    //    ***************************************************************
    // *SHUT-IN TIME (BUILDUP) CALCULATIONS
*
    //    ***************************************************************
    //    *COMPUTE THE EQUIVALENT CONSTANT RATE PRODUCING TIME TO BE      *
    // *USED IN PRESSURE BUILDUP CALCULATIONS (PRESSURE BUILDUP     *
    //    *IS ASSUMED TO FOLLOW A PRODUCTION PERIOD OF EQUIVALENT              *
    // *PRODUCING TIME AT A CONSTANT RATE OF QTP EQUAL TO THE LAST  *
    // *RATE AT THE END OF THE PRODUCTION PERIOD)
*
    // ***************************************************************
                TPN=24.*QGCN/QTPN;
    //    ***************************************************************
    // *COMPUTE THE BUILDUP PRESSURES
        *
    //    ***************************************************************
                        DTN=0;
                        CH2=0.01;

    start2:
```

```
                    TN=TPN+DTN;
          PBUP(DMWS,N,V,TPN,TN,QTPN,TAV,CWS,SK,RW,RE,H,PERM,ETA,IFR);
    //
                    if (DMWS <= 0){
    //                      cout<<"***DMWS<=0***"<<endl;
                            T=TT;
                            TP=TPT;
                            DT=DTT;
                            PCAV=PCAVT;
                            PCMAX=PCMAXT;
                            PWS=PWST;
                            QTP=QTPT;
                            QGC=QGCT;
                            PSPIN=PSPIN-
2.*1422.*(TAV+460.)*ETA*QGC/24./RE/RE/PERM/H;
                            PIN=FLAGR(PSP,PRE,PSPIN,1,NPRES,IERR);
                            goto end;
                    }
                    PSPWSN=PSPIN-DMWS;
                    PWSN=FLAGR(PSP,PRE,PSPWSN,1,NPRES,IERR);
                    DPN=PWSN-PCMAXN;
    //              cout<<"DPN="<<DPN<<"DTN="<<DTN<<"TPN="<<TPN<<endl;
    //    ****************************************************************
    //  *CHECK TO SEE IF PRODUCING TIME IS TOO LONG
*
    //  *(PCMAX SHOULD BE LESS THAN PWS)
*
    //  *IF YES, REDUCE PRODUCING TIME
          *
    //  ****************************************************************
                    if(DPN >= 0.){
                     if(DTN == 0.){
                          if(TPCP >= CH1*10.)
                               CH1=CH1*10.;
                          TPCP=TPCP+0.1*CH1;
                          goto start1;
                     }
                     else{
```

```
                                        DQCN=100.*(QGCN-QGCT)/QGCN;
                                                if(DQCN >= 1.){
                                                        TT=TN;
                                                        TPT=TPN;
                                                        DTT=DTN;
                                                        PCAVT=PCAVN;
                                                        PCMAXT=PCMAXN;
                                                        PWST=PWSN;
                                                        QTPT=QTPN;
                                                        QGCT=QGCN;
                                                }
                                                if(TPCP >= CH1*10.)
                                                        CH1=CH1*10.;
                                                TPCP=TPCP+0.1*CH1;
                                                goto start1;
                                        }
                                }
                                else{
                                        if(DTN >= CH2*10.)
                                                CH2=CH2*10.;
                                        DTN=DTN+0.1*CH2;
                                        goto start2;
                                }

        end:
        ;
        //      ************************************************************
        //      *OUTPUT RESULTS
                                *
        //      ************************************************************
                /*                      TIME=TIME+TP;
                                        QGCTOT=QGCTOT+QGC;
                                        QTPP=QTP;
                                        cout<<"TIME="<<TIME<<"          "<<"QGCTOT="<<QGCTOT<<"
"<<"QTPP="<<QTPP<<endl;

outfil<<setw(10)<<TIME<<setw(10)<<QGCTOT<<setw(10)<<QTPP<<endl;
                                        TIME=TIME+DT;
```

```
                                    QTPP=0;
                                    cout    <<"TIME="<<TIME<<"       "<<"QGCTOT="<<QGCTOT<<"
"<<"QTPP="<<QTPP<<endl;

outfil<<setw(10)<<TIME<<setw(10)<<QGCTOT<<setw(10)<<QTPP<<endl;
     //
                                    cout<<"DT="<<DT<<"   "<<"TP="<<TP<<endl;
                                    cout
<<"*****************************************************"<<endl;
     */
     //


     //
     }

     //    ************************************************************
     //    *                SUBROUTINE PBUP                                    *
     //    *
                                    *
     //    *PROGRAM COMPUTES BUILDUP PRESSURES FOLLOWING CONSTANT          *
     //    *PRESSURE PRODUCTION USING LAPLACE DOMAIN SOLUTIONS AND
*
     //    *NUMERICAL INVERSION
                    *
     //    ************************************************************
     void PBUP(double &DMWS, int N,double V[],double TP, double T, double QTP, double TAV,
                    double CWS, double SK, double RW, double RE, double H,
                    double PERM, double ETA, int IFR){


          double A, ARG;
        double DLOGTW=.6931471805599453;
          double FREG, FA1, FA2, TTP;
          double TTPAD;
          int IFR1;


          PI=4*atan(1);
     //
     //
     //
```

```
   FA1=0.;
   A=DLOGTW/T;
     for (int I=0; I<N; ++I){
                ARG=A*(I+1);
      FA1=FA1+V[I+1]*PU(ARG,RE,RW,CWS,SK,ETA,IFR,PERM,H,TAV,QTP);
     }
//
    FA1=A*FA1;
    if(fabs(FA1) <= pow(10,-38)){
                FA1=0.;
      }
//
                TTP=T-TP;
//
     if(TTP > 0){
                TTPAD=ETA*TTP/PI/RE/RE;
                IFR1=0;
                FREG=TTPAD-0.1;
                if(FREG >=0 )
                        IFR1=1;
//
      FA2=0;
                A=DLOGTW/TTP;
        for (int I=0; I<N+1; ++I){
                   ARG=A*(I+1);
        FA2=FA2+V[I+1]*PU(ARG,RE,RW,CWS,SK,ETA,IFR,PERM,H,TAV,QTP);
                }

      FA2=A*FA2;
      if(fabs(FA2) <= pow(10,-38)){
                   FA2=0.;
                }
     }
//
     else
       FA2=0.;
//
```

```
      DMWS=FA1-FA2;
        if (DMWS <= 0.)
                DMWS=0.;
//
      }
//    ************************************************************
double PU (double S, double RE, double RW, double CWS, double SK,
                double ETA, int IFR, double PERM, double H, double TAV, double QTP){
//
       double P, ARG1, ARG2, TOP, BOT, PU;
//
     ARG1=sqrt(S/ETA)*RW;
       ARG2=sqrt(S/ETA)*RE;
//
       if (IFR == 0)
       P=BESSK0(ARG1)/S/ARG1/BESSK1(ARG1)+SK/S;
       else
         P=((BESSK1(ARG2)*BESSI0(ARG1)+BESSI1(ARG2)*BESSK0(ARG1))/
        S/ARG1/(BESSI1(ARG2)*BESSK1(ARG1)-BESSK1(ARG2)*BESSI1(ARG1)))
         +SK/S;

//
      P=(1422.*(TAV+460.)/PERM/H)*P;
//
      TOP=P;
      BOT=1+24.*CWS*S*S*P;
   PU=QTP*(TOP/BOT);
//
return PU;

}
//    ************************************************************
void QCPAV (double &QGC, double &QTP, double &PCMAX, double &PCAVN, double &CG,
                        double &ETA, int &IFR, double &DPSPC, int N, double V[], double SK,
                        double RW, double RE, double PERM, double PHI, double H, double VISG,
                        double TAV, double PCAVO, double SGG, int MCODE, double WM, double
TID,
```

```
                    double EPSD, double TP, double GLR, double SGL, double FL, double VPAV,
                    double BL, double PP, double PTMIN, double P1H, double AT, double AA,
                    double D, double PSPIN, double PRE[], double PSP[], int NPRES){
//

      int ICO, ICN, I;
      double DIF,PSPC,TPAD,FREG, PI;
      double PCAVO2, PCAVO22;

      PI=4.*atan(1.);
      I=0;
      ICO=0;
      ICN=0;
      PCAVO2=0.;

start:

       PSPC=FLAGR (PRE,PSP,PCAVO,1,NPRES,IERR);
       DPSPC=PSPIN-PSPC;
       PROP (CG,VISG,TAV,PCAVO,SGG,MCODE);
//    ************************************************************
//    *COMPUTE ETA AND DIMENSIONLESS PRODUCING TIME BASED ON          *
//  *THE DRAINAGE AREA.
                *
//  *CHECK FOR FLOW REGIMES (IFR=0; INFINITE ACTING,                  *
//  *IFR=1; BOUNDARY DOMINATED)
                *
//    ************************************************************
       ETA=0.0002637*PERM/PHI/(CG)/VISG;
       TPAD=ETA*TPCP/PI/RE/RE;
       IFR=0;
       FREG=TPAD-0.1;
       if(FREG >=0.)
         IFR=1;

//
       CPROD (QGC,QTP,N,V,TP,SK,RW,RE,ETA,IFR);
       QGC=DPSPC*PERM*H*QGC/24./1422./(TAV+460.);
```

```
      QTP=DPSPC*PERM*H*QTP/1422./(TAV+460.);

//
      PCAVSUB (PCAVN,PCMAX,TAV,PCAVO,SGG,MCODE,WM,TID,EPSD,QGC,GLR,SGL,
            FL,VPAV,BL,PP,PTMIN,P1H,AT,AA,D);

      DIF=PCAVN-PCAVO;

      if(fabs(DIF) > 0.1){
      if(I == 0){
       PCAVO2=PCAVO;
              if(DIF > 0.)
        PCAVO=PCAVO+0.17;
              else
        PCAVO=PCAVO-0.13;
       I=1;
              goto start;
      }
      else{
       if(DIF > 0.)
        ICN=1;
       else
               ICN=-1;

       if((ICO/ICN) < 0){
        PCAVO22=PCAVO;
               PCAVO=(PCAVO+PCAVO2)/2.;
               PCAVO2=PCAVO22;
              }
              else{
               PCAVO2=PCAVO;
        if(DIF > 0.)
         PCAVO=PCAVO+0.17;
        else
         PCAVO=PCAVO-0.13;
              }
```

```
        }
          ICO=ICN;
                goto start;
        }
//
        PCMAX=PCAVN*(2*(AA+AT))/(2*AA+AT);
//
        return;
                        }
//    ************************************************************
//    *                SUBROUTINE CPROD                                    *
// *
                *
//    *PROGRAM COMPUTES CUMULATIVE PRODUCTION FOR A GIVEN PRODUCING*
//    *TIME AND CONSTANT PRODUCTION PRESSURE USING THE LAPLACE          *
//    *DOMAIN SOLUTION AND NUMERICAL INVERSION
*
//    ************************************************************
void CPROD(double &QGC, double &QG, int N, double V[], double T, double SK, double RW,
                double RE, double ETA, int IFR) {
//
   double A, ARG;
   double DLOGTW=.6931471805599453;
//
   QGC=0;
     QG=0;
   A=DLOGTW/T;
     for (int I=0; I < N; ++I){
                ARG=A*(I+1);
       QGC=QGC+V[I+1]*Q(ARG, RE, RW, SK, ETA, IFR);
       QG=QG+V[I+1]*ARG*Q(ARG, RE, RW, SK, ETA, IFR);
     }

    QGC=A*QGC;
      QG=A*QG;
    if(fabs(QGC)<= pow(10,-38))
                QGC=0.;
```

```
      if(fabs(QG)<= pow(10,-38))
                  QG=0.;

//

}
//    ***************************************************************
      double Q(double S, double RE, double RW, double SK, double ETA, int IFR){
//
      double P, ARG1, ARG2, Q;
   ARG1=sqrt(S/ETA)*RW;
   ARG2=sqrt(S/ETA)*RE;
//
      if(IFR == 0)
     P=BESSK0(ARG1)/S/ARG1/BESSK1(ARG1)+SK/S;
      else
       P=((BESSK1(ARG2)*BESSI0(ARG1)+BESSI1(ARG2)*BESSK0(ARG1))/
      S/ARG1/(BESSI1(ARG2)*BESSK1(ARG1)-BESSK1(ARG2)*BESSI1(ARG1)))
      +SK/S;

     Q=(1/S/S/S)/P;
//
return Q;
      }
//    ***************************************************************
      void PROP(double &CG, double &VISG, double TAV, double P, double SGG, int MCODE){
//
      double DZ, P1, Z1, Z;

      PI=4*atan(1);


   VISG=GASVIS (TAV,SGG,P,IERR);

   Z=ZFAC (TAV,P,SGG,MCODE,IERR);
      P1=P*0.9;
```

```
      Z1=ZFAC (TAV,P1,SGG,MCODE,IERR);
        DZ=(Z-Z1)/(P-P1);

        CG=(1./P)-(DZ/Z);

    //

        }
    //  ************************************************************
    void PCAVSUB(double &PCAVN, double &PCMAX, double TAV, double PCAVO, double SGG,
                          int MCODE, double WM, double TID, double EPSD, double QGC, double GLR,
                          double SGL, double FL, double VPAV, double BL, double PP,
                          double PTMIN, double P1H, double AT, double AA, double D){

    //

        double Z, RHOG, VISG, REG, FG, XLMAX, XL, P1F, RK, PL;

        PI=4.*atan(1.);

    //

      Z=ZFAC(TAV,PCA VO,SGG,MCODE,IERR);
      RHOG=PCAVO*WM/Z/10.732/(TAV+460.);

      VISG=GASVIS(TAV,SGG,PCAVO,IERR);
        REG=3.527*RHOG*VPAV*TID/VISG;

        FG=FC(REG,EPSD);

        XLMAX=PI*TID*TID*D/4./144./5.615;
        XL=QGC*1000./GLR;
        if (XL > XLMAX)
                XL=XLMAX;

        P1F=0.433*SGL*FL*VPAV*VPAV*BL/772.8/TID;

        RK=RHOG*FG*VPAV*VPAV/TID/772.8/PCAVO;
        PL=PP+PTMIN+(P1H+P1F)*XL+14.7;
```

```
      PCAVN=(1.+AT/2./AA)*PL*(1.+D*RK);

      PCMAX=PCAVN*(2.*(AA+AT)/(2.*AA+AT));

//

      }
//    ************************************************************
//
      double BESSK1(double X){
//
//    This program calculates K1(x). The program is taken from Numerical
//   Recipies by Press et al.
//
//   Subprograms Called:
//          BESSI1
//
      double BESSK1, Y;
      double P1, P2, P3, P4, P5, P6, P7;
      double Q1, Q2, Q3, Q4, Q5, Q6, Q7;

      P1=1.;
      P2=0.15443144;
      P3=-0.67278579;
      P4=-0.18156897;
      P5=-0.01919402;
      P6=-0.00110404;
      P7=-0.00004686;
//
      Q1=1.25331414;
      Q2=0.23498619;
      Q3=-0.03655620;
      Q4=0.01504268;
      Q5=-0.00780353;
      Q6=0.00325614;
      Q7=-0.00068245;
//
      if (X <= 0.){
```

```
        //cout<<"BAD ARGUMENT IN BESSK1"<<endl;
        BESSK1=0.;
        return BESSK1;
      }
//
      if (X <= 2.){
        Y=X*X/4.;
        BESSK1=(log(X/2.)*BESSI1(X))+(1./X)*(P1+Y*(P2+
                Y*(P3+Y*(P4+Y*(P5+Y*(P6+Y*P7))))));}
      else{
        Y=2./X;
        BESSK1=(exp(-X)/sqrt(X))*(Q1+Y*(Q2+Y*(Q3+
                Y*(Q4+Y*(Q5+Y*(Q6+Y*Q7))))));}
//
      return BESSK1;
      }
//    ****************************************************************
double BESSI1(double X){
//    ****************************************************************
//  *This program calculates I1(x). The program is taken from    *
//  *Numerical Recipies by Press et al.                                *
//    ****************************************************************
      double BESSI1, Y, AX;
      double P1, P2, P3, P4, P5, P6, P7;
      double Q1, Q2, Q3, Q4, Q5, Q6, Q7,Q8,Q9;

      P1=0.5;
      P2=0.87890594;
      P3=0.51498869;
      P4=0.15084934;
      P5=0.02658733;
      P6=0.00301532;
      P7=0.00032411;
//
      Q1=0.39894228;
      Q2=-0.03988024;
      Q3=-0.00362018;
```

```
      Q4=0.00163801;
   Q5=-0.01031555;
      Q6=0.02282967;
      Q7=-0.02895312;
      Q8=0.01787654;
      Q9=-0.00420059;
//
   if (fabs(X) <= 3.75){
       Y=pow(X/3.75,2);
       BESSI1=X*(P1+Y*(P2+Y*(P3+Y*(P4+Y*(P5+Y*(P6+Y*P7))))));}
      else{
       AX=fabs(X);
       Y=3.75/AX;
       BESSI1=(exp(AX)/sqrt(AX))*(Q1+Y*(Q2+Y*(Q3+Y*(Q4+
                  Y*(Q5+Y*(Q6+Y*(Q7+Y*(Q8+Y*Q9))))))));}
//
      return BESSI1;
}
//    ************************************************************
      double BESSK0(double X){
//    ************************************************************
// *This  program calculates K0(x). The program is taken from    *
// *Numerical Recipies by Press et al.                                                    *
//    *
                                    *
// *Subprograms Called: BESSI0
       *
//    ************************************************************
      double BESSK0, Y;
      double P1, P2, P3, P4, P5, P6, P7;
      double Q1, Q2, Q3, Q4, Q5, Q6, Q7;


      P1=-0.57721566;
      P2=0.42278420;
      P3=0.23069756;
      P4=0.03488590;
      P5=0.00262698;
```

```
      P6=0.00010750;
      P7=0.0000074;

      Q1=1.25331414;
      Q2=-0.07832358;
      Q3=0.02189568;
      Q4=-0.01062446;
      Q5=0.00587872;
      Q6=-0.00251540;
      Q7=0.00053208;
//
      if(X <=0.){
            //cout<<"BAD ARGUMENT IN BESSK0"<<endl;
            BESSK0=0.;
            return BESSK0;
      }
//
      if (X <=2.){
            Y=X*X/4.;
            BESSK0=(-log(X/2.)*BESSI0(X))+(P1+Y*(P2+Y*(P3+
                  Y*(P4+Y*(P5+Y*(P6+Y*P7))))));}
      else{
            Y=(2./X);
            BESSK0=(exp(-X)/sqrt(X))*(Q1+Y*(Q2+Y*(Q3+
            Y*(Q4+Y*(Q5+Y*(Q6+Y*Q7))))));}
//
      return BESSK0;
}

//
//    ************************************************************
      double BESSI0(double X){
//    ************************************************************
//  *This program calculates I0(x). The program is taken from    *
//  *Numerical Recipies by Press et al.                                    *
//    ************************************************************
      double BESSI0, Y, AX;
```

```
        double P1, P2, P3, P4, P5, P6, P7;
        double Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9;


        P1=1;
        P2=3.5156229;
        P3=3.0899424;
        P4=1.2067492;
        P5=0.2659732;
        P6=0.0360768;
        P7=0.0045813;

        Q1=0.39894228;
        Q2=0.01328592;
        Q3=0.00225319;
        Q4=-0.00157565;
        Q5=0.00916281;
        Q6=-0.02057706;
        Q7=0.02635537;
        Q8=-0.01647633;
        Q9=0.00392377;

        if (fabs(X) < 3.75){
                Y=pow(X/3.75,2);
                BESSI0=P1+Y*(P2+Y*(P3+Y*(P4+Y*(P5+Y*(P6+Y*P7)))));}
        else{
                AX=fabs(X);
                Y=3.75/AX,
                BESSI0=(exp(AX)/sqrt(AX))*(Q1+Y*(Q2+Y*(Q3+Y*(Q4
                +Y*(Q5+Y*(Q6+Y*(Q7+Y*(Q8+Y*Q9)))))))));}
//
        return BESSI0;
}

//      ************************************************************
void STEHFESTV(double VV[], int NN){
```

```
double G[52],H[26];
double FI, SN;
   int NH, K, kc1, K1, K2, i, ic1;

   VV[0]=0.;
   H[0]=0.;
   G[0]=0.;
G[1]=1.;
NH=NN/2;
   for (i=0; i<NN-1; ++i){
              ic1=i+2;
              G[ic1]=G[ic1-1]*(ic1);
   }

H[1]=2.0/G[NH-1];

   for (i=0; i<NH-1; ++i){
             ic1=i+2;
  FI=ic1;
  if(ic1==NH)
                       H[ic1]=pow(FI,NH)*G[2*ic1]/(G[ic1]*G[ic1-1]);
             else
  H[ic1]=pow(FI,NH)*G[2*ic1]/(G[NH-ic1]*G[ic1]*G[ic1-1]);
  }

SN=2*(NH-NH/2*2)-1;

   for (i=0; i<NN; ++i){
             ic1=i+1;
              VV[ic1]=0.0;
  K1=(ic1+1)/2;
  K2=ic1;
  if(K2 > NH)
                        K2 = NH;

  for(K=0; K < K2-K1+1; ++K){
               kc1=K+K1;
```

```
                        if((2*kc1-ic1)==0)
                                VV[ic1]=VV[ic1]+H[kc1]/G[ic1-kc1];
          else if (ic1==kc1)
                                VV[ic1]=VV[ic1]+H[kc1]/G[2*kc1-ic1];
                        else
                                VV[ic1]=VV[ic1]+H[kc1]/(G[ic1-kc1]*G[2*kc1-ic1]);
                 }

        VV[ic1]=SN*VV[ic1];
                 SN=-SN;
        }
       }
//  ***************************************************************
double FC(double RE, double ED){
//
      double FCI, arg, DEN, FF, DIFF, FC;
//
      if(RE < 2300.)
                FC=16./RE;
      else{
//    ***************************************************************
//  *CALCULATE MOODY FRICTION FACTOR WITH JAIN EQUATION FOR FIRST*
//    *GUESS FROM COLEBROOK EQUATION
                *
//    ***************************************************************
                arg=1.14-2.*log10(ED+21.25/pow(RE, 0.9));
                FCI=1.0/arg/arg;
//    ***************************************************************
//  *SET COUNTER. COLEBROOK EQUATION IS ITERATIVE. IF CONVERGENCE*
//  *IS NOT ATTAINED IN 10 ITERATIONS AN INFINITE LOOP WILL          *
//    *PROBABLY OCCUR. SET FRICTION FACTOR EQUAL TO THE VALUE          *
//    *DETERMINED IN THE 10TH ITERATION AND USE WITH CAUTION           *
//    ***************************************************************
                DIFF=1.0;
                int i=0;
                do {
                        i=i+1;
```

```
                    DEN=1.14-2.0*log10(ED+9.34/(RE*sqrt(FCI)));
                    FF=pow(1.0/DEN, 2);
                    DIFF=fabs(FCI-FF);
                    FCI=(FCI+FF)/2.0;
            } while(DIFF > 0.00001 && i < 10);

            FF=FCI;

            FC=FF/4.0;
    }
return FC;
}
//  ********************************************************************
//
//  This subroutine calculates viscosity of hydrocarbon gases from
//  the following correlation.  The English system of units is used
//  in the calculation.
//
//  - Viscosity of hydrocarbon gases calculation:
//    The Lee et al correlation is used.
//
//
//                    REFERENCES
//                    ----------
//
//  1.  Brill, J. P. and Beggs, H. D.:  Two-Phase Flow in Pipes
//        (Feb. 1984) 2-58 thru. 2-63.
//  2.  Lee, A. L., et al.:  "The Viscosity of Natural Gases,"
//        Transactions, AIME (Aug. 1966) 997-1000.
//
//  ********************************************************************
//
//                  SUBPROGRAM CALLED
//                  -----------------
//
//  ZFAC = This subroutine calculates gas compressibility factor.
//
```

```
//
//                    VARIABLE DESCRIPTION
//                    --------------------
//
//     DENSI = Gas density. (gm/cc)
//     IERR  = Error code. (0=OK, 1=input variables out of range,
//           2=extrapolation of correlation occuring)
//     IOERR = Output file for error messages when input values
//           passed to the subroutine are out of range.
//    *P     = Pressure. (psia)
//    *SGFG  = Specific gravity of free gas.
//    *T     = Temperature. (deg-F)
//     TABS  = Absolute temperature. (deg-R)
//     VISG  = Gas viscosity. (cp)
//     W     = Molecular weight.
//     Z     = Real gas compressibility factor.
//
//     AK,X,Y = Dummy variables.
//     (* Indicates input variables)
//
//    ******************************************************************
      double GASVIS (double T, double SGFG,double P, int IERR){


      double TABS, AK, W, XX, YY, ZFACTOR, DENSI, GASVIS;
      int MCODE;

//    ************************************
//    Check input variables for valid range.
//    ************************************

      IERR=0;
        if (T < 0.0 || T > 400.0){
        //cout<<"GASVIS:  Illegal input value for T"<<endl;
         IERR=1;}
        if (SGFG < 0.20 || SGFG > 1.7){
        //cout<<"GASVIS:  Illegal input value for SGFG"<<endl;
```

```
                IERR=1;}
        if(P < 0.0){
                //cout<<"GASVIS:  Illegal input value for P"<<endl;
                IERR=1;}
        if(IERR==1){
                GASVIS = 0.;
                return GASVIS;}
//
//    ---------------------------------------------------
//    Check input variables for valid correlation range.
//    ---------------------------------------------------
//
     //  if (T < 100.0 || T > 340.0)
     //          cout<<"T is out of range for the Lee correlation in GASVIS\n"<<
     //                  "and extrapolation is occuring\n"<<endl;
  // else
                //if (P < 100.0 || P > 8000.0)
                //cout<<"P is out of range for the Lee correlation in GASVIS\n"<<
                //          "and extrapolation is occuring\n"<<endl;
//
//    **********************
//    End of validity check.
//    **********************
//
                TABS=T+460.;
                W=SGFG*29.;
                AK=(9.4+.02*W)*(pow(TABS,1.5))/(209.+19.*W+TABS);
                XX=3.5+(986./TABS)+.01*W;
                YY=2.4-.2*XX;
//
//    ----------------------
//    Calculate gas density.
//    ----------------------
//
                MCODE=0;
          ZFACTOR=ZFAC(T,P,SGFG,MCODE,IERR);
          DENSI=P*W/(10.72*ZFACTOR*TABS*62.4);
```

```
//
//    ------------------------
//    Calculate gas viscosity.
//    ------------------------
//
      VISG=AK*exp(XX*pow(DENSI,YY))/10000;
//
      GASVIS=VISG;
//
      return GASVIS;
      }
//
//    ********************************************************************
//
//    This subroutine calculates gas compressibility factor from the
//    following correlations.  The English system of units is used in
//    the calculation.
//
//    - Gas compressibility factor correlations used are selected by
//      MCODE to be either:
//         0:  The Hall and Yarborough correlation for curve fitting
//             the Standing-Katz reduced pressure-reduced temperature
//             Z-Factor chart.
//         1:  The standing modification to the Brill and Beggs
//             correlation for curve-fitting the Standing-Katz reduced
//             pressure-reduced temperature Z-Factor chart.
//         2:  The Dranchuk, Purvis and Robinson correlation for curve
//             fitting the Standing-Katz reduced pressure-reduced tem-
//             perature Z-Factor chart.
//       or 3:  The Gopal correlation for curve fitting the Standing-
//             -Katz reduced pressure-reduced temperature Z-Factor
//             chart.
//
//
//                          REFERENCES
//                          ----------
//
```

```
//    1.  Brill, J. P. and Beggs, H. D.:  Two-Phase Flow in Pipes
//          (Feb. 1984) 2-33 thru. 2-47.
//    2.  Dranchuk, P. M., Purvis, R. A. and Robinson, D. B.:  "Computer
//          Calculation of Natural Gas Compressibility Factors Using
//          the Standing and Katz Correlation," Institute of Petroleum
//          Technical Series, NO. IP74-008, 1974.
//    3.  Gopal, V. N.:  "Gas Z-Factor Equations Developed for Computer,"
//          Oil and Gas Journal (Aug. 8, 1977) 58-60.
//    4.  Hall, K. R. and Yarborough. L.:  "A New Equation of State for
//          Z-Factor Calculations,"  Oil and Gas Journal (June 18,
//          1973) 82-92.
//    5.  Standing, M. B.:  " Volumetric and Phase Behavior of Oil Field
//          Hydrocarbon Systems", Society of Petroleum Engineers
//          (8th Printing, 1977) 121-127.
//    6.  Standing, M. B. and Katz, D. L.:  "Density of Natural Gases,"
//          Transactions, AIME, 196. (1942) 140-149.
//    7.  Yarborough, L. and Hall, K. R.:  "How to Solve Equation of State
//          for Z-Factors,"  Oil and Gas Journal (Feb. 18, 1974)
//          86-88.
//
//    ***************************************************************
//
//                      VARIABLE DESCRIPTION
//                      --------------------
//
//     DENR  = Reduced density.
//     IERR  = Error code. (0=OK, 1=input variables out of range,
//          2=extrapolation of correlation occuring)
//     MCODE = Z-Factor correlation selection parameter:
//               0 = Hall and Yarborough
//               1 = Standing
//               2 = Dranchuk, Purvis and Robinson
//               3 = Gopal.
//    *P     = Pressure. (psia)
//     PC    = Critical pressure. (psia)
//     PR    = Reduced pressure.
//     RT    = Inverse of reduced temperature.
```

```
//     *SGFG  = Specific gravity of free gas.
//     *T     = Temperature. (deg-F)
//      TC    = Critical temperature. (deg-R)
//      TR    = Reduced temperature.
//      Y     = Data for equation coefficients.
//      Z     = Real gas compressibility factor.
//
//     A,B,//,D,E,F,G,DFDY,FN,I,J,K = Dummy variables.
//      (* Indicates input variables)
//
//     ******************************************************************
//
double ZFAC (double T, double P, double SGFG, int MCODE, int IERR){


        double TC, PC, TR, PR, ZFAC;

//     ************************************
//     Check input variables for valid range.
//     ************************************
        IERR=0;
        TC=0.;

        if (T < 0. || T > 800.){
                //cout<<"ZFAC:  Illegal input value for T in ZFAC"<<endl;
        IERR=1;
        }
        if(P < 0. || P > 10000.){
                //cout<<"ZFAC:  Illegal input value for P in ZFAC"<<endl;
                IERR=1;
        }
        if(SGFG < 0.55 || SGFG > 1.5){
                //cout<<"ZFAC:  Illegal input value for SGFG in ZFAC"<<endl;
                IERR=1;
        }
        if(MCODE < 0 || MCODE > 3){
                //cout<<"ZFAC:  Illegal input value for MCODE in ZFAC"<<endl;
```

```
                IERR=1;
        }
//    **********************
//    End of validity check.
//    **********************

        if(IERR == 1)
                ZFAC = 0.;
        else{
//
//    ------------------------------------------------------
//    Calculate critical and reduced temperature and pressure.
//    ------------------------------------------------------

      TC=169.0+314.0*SGFG;
      PC=708.75-57.5*SGFG;
      TR=(T+460.0)/TC;
      PR=P/PC;
//
//    ----------------------------
//    Select Z-Factor correlation.
//    ----------------------------
//
                if (MCODE == 0)
          hallyarbcorr (ZFAC, PR, TR);
                else if (MCODE == 1)
                        standingcorr (ZFAC, PR, TR);
       else if (MCODE == 2)
          dranchukcorr (ZFAC, PR, TR);
                else
                        gopalcorr (ZFAC, PR, TR);
        }
//
      return ZFAC;
}
//
//    *******************************
```

```
//    HALL AND YARBOROUGH CORRELATION
//    ********************************
//
//    ----------------------------------------------------------------

      void hallyarbcorr (double& Z, double PR, double TR){

//    If reduced temperature is less than 1.01, calculate a Z-Factor
//    for a reduced temperature value of 1.0.
//    ----------------------------------------------------------------
        double RT, A, B, C, D, F, DENR, DFDY;

              if (TR > 1.01)
                    RT=1./TR;
              else
                    RT=1.;
//    ---------------------------------------
//    Calculate temperature dependent terms.
//    ---------------------------------------
//
       A=0.06125*RT*exp(-1.2*pow(1.-RT,2));
       B=RT*(14.76-9.76*RT+4.58*RT*RT);
       C=RT*(90.7-242.2*RT+42.4*RT*RT);
       D=2.18+2.82*RT;

//    ------------------------------------------------------------------
//    Calculate reduced density, DENR, using the Newton-Raphson method.
//    ------------------------------------------------------------------

              DENR=.001;

              for(int j=0; j < 25; ++j){
                    if (DENR > 1)
                          DENR=.6;

                    if( DENR <= 0){
                     gopalcorr(Z, PR, TR);
```

```
                    return;
                }
                F=-A*PR+(DENR+DENR*DENR+pow(DENR,3)-pow(DENR,4))/pow(1.-
DENR,3)
                    -B*DENR*DENR+C*pow(DENR,D);

                if(fabs(F)<=.0001){
    //      ------------------
    //      Calculate Z-factor.
    //      ------------------

                    Z=A*PR/DENR;
                    return;
                }
    //  ------------------------------------------------------
    //  If convergence is not obtained in 25 iterations, set Z=1.0
    //  and return.
    //  ------------------------------------------------------

                if (j < 24){
                    DFDY=(1.+4.*DENR+4.*DENR*DENR-
4.*pow(DENR,3)+pow(DENR,4))
                        /pow((1.-DENR),4)-2.*B*DENR+D*C*pow(DENR,(D-1.));
                    DENR=DENR-F/DFDY;}
                else{
                    Z=1;
                    return;}
            }
    return;
        }

    //  ********************
    //  STANDING CORRELATION
    //  ********************
        void standingcorr (double& Z, double PR, double TR){

            double A, B, C, D, E, F, G;
```

```
            if (TR < 1.2 || TR > 2.4) {
                gopalcorr(Z, PR, TR);
                return;
            }
        A=1.39*pow(TR-.92,0.5)-.36*TR-.101;
        B=(.62-.23*TR)*PR;
        C=(.066/(TR-.86)-.037)*pow(PR,2);
        D=(.32/(pow(10,(9.*(TR-1.)))))*pow(PR,6);
        E=B+C+D;
        F=(.132-.32*log10(TR));
        G=pow(10,(.3106-.49*TR+.1824*pow(TR,2)));
//
//    ------------------
//    Calculate Z-Factor.
//    ------------------
//
        Z=A+(1.-A)*exp(-E)+F*pow(PR,G);
        return;
      }

//    ****************************************
//    DRANCHUK, PURVIS AND ROBINSON CORRELATION
//    ****************************************

      void dranchukcorr(double& Z, double PR, double TR)
      {
//
      double A, B, C, D, E, F, G, DENR, DFDY, FN;
//    ------------------------------------------------------------
//    Calculate Benedict-Webb-Rubin Equation of State Coefficients.
//    ------------------------------------------------------------
//
        A=0.06423;
        B=0.5353*TR-0.6123;
        C=0.3151*TR-1.0467-0.5783/pow(TR,2);
        D=TR;
        E=0.6816/pow(TR,2);
```

```
        F=0.6845;
        G=0.27*PR;
//
//  ------------------------------------------------------
//  Guess initial value for reduced density and use the Newton-
//  Raphson iteration method to determine reduced density.
//  ------------------------------------------------------
//
            DENR=0.27*PR/TR;
            for(int i=0; i<25; ++i){

    FN=A*pow(DENR,6)+B*pow(DENR,3)+C*pow(DENR,2)+D*DENR+E*pow(DENR,3)
                        *(1.0+F*pow(DENR,2))*exp(-F*pow(DENR,2))-G;

                    if (fabs(FN) <= 0.0001){

//  -------------------
//  Calculate Z-Factor.
//  -------------------
//
          Z=0.27*PR/(DENR*TR);
          return;
                    }

//  ------------------------------------------------------------
//  If convergence is not obtained in 25 iterations, set Z=1.0
//  and return.
//  ------------------------------------------------------------
//
                    if (i < 24) {

DFDY=6.0*A*pow(DENR,5)+3.0*B*pow(DENR,2)+2.0*C*DENR+D+E*pow(DENR,2)
          *(3.0+F*pow(DENR,2))*(3.0-2.0*F*pow(DENR,2))*exp(-F*pow(DENR,2));
                        DENR=DENR-FN/DFDY;}
                    else{
                        Z=1.;
                        return;}
            }
```

```
                return;
        }
//
//    *****************
//    GOPAL CORRELATION
//    *****************
//
void gopalcorr(double& Z, double PR, double TR)
{

        double Y[49];
        int i, j, k;

          Y[0]=0;
          Y[1]=1.6643;
          Y[2]=-2.2114;
          Y[3]=-.3647;
          Y[4]=1.4385;
          Y[5]=.5222;
          Y[6]=-.8511;
          Y[7]=-.0364;
          Y[8]=1.0490;
          Y[9]=.1391;
          Y[10]=-.2988;
          Y[11]=.0007;
          Y[12]=.9969;
          Y[13]=.0295;
          Y[14]=-.0825;
          Y[15]=.0009;
          Y[16]=.9967;
          Y[17]=-1.3570;
          Y[18]=1.4942;
          Y[19]=4.6315;
          Y[20]=-4.7009;
          Y[21]=.1717;
          Y[22]=-.3232;
          Y[23]=.5869;
```

```
    Y[24]=.1229;
    Y[25]=.0984;
    Y[26]=-.2053;
    Y[27]=.0621;
    Y[28]=.8580;
    Y[29]=.0211;
    Y[30]=-.0527;
    Y[31]=.0127;
    Y[32]=.9549;
    Y[33]=-.3278;
    Y[34]=.4752;
    Y[35]=1.8223;
    Y[36]=-1.9036;
    Y[37]=-.2521;
    Y[38]=.3871;
    Y[39]=1.6087;
    Y[40]=-1.6635;
    Y[41]=-.0284;
    Y[42]=.0625;
    Y[43]=.4714;
    Y[44]=-.0011;
    Y[45]=.0041;
    Y[46]=.0039;
    Y[47]=.0607;
    Y[48]=.7927;

  if (PR < 0.199)
   Z=1.;
else {
        if (PR > 5.4)
    Z=PR*pow(.711+3.66*TR,-1.4667)-1.637/(.319*TR+.522)+2.071;
        else {
            i=1;
            if (PR > 1.2){
                if(PR > 1.4 || TR < 1.08 || TR > 1.19) {
                    if (PR <= 2.8)
              i=2;
```

```
            else
                i=3;

                                }
                        }
                        k=4;
        if (TR <= 2.0)
                                k=3;
        if (TR <= 1.4)
                                k=2;
        if (TR <= 1.2)
                                k=1;
        if (TR <= 1.0){
                                Z=1.;
                                return;
                        }
                        j= 16*i+4*k-19;

                        Z=PR*(Y[j]*TR+Y[j+1])+Y[j+2]*TR+Y[j+3];
                }
    }
return;
}

//    ****************************************************************
//
//    This subroutine uses the Lagrange Formula to evaluate the
//    interpolating polynomial of degree IDEG for argument XARG using
//    the data values X(MIN).....X(MAX) and Y(MIN).....Y(MAX) where
//    MIN = MAX-IDEG. The X(I) values are not necessarily evenly
//    spaced and can be in either increasing or decreasing order.
//
//    Interpolation routine similar to      in 'Applied Numerical
//    Methods' by Carnahan, Luther and Wilkes.
//
//
//                        REFERENCE
```

```
//                        ---------
//
//    1.  Carnahan, Luther and Wilkes.:  Applied Numerical Methods,
//          John Wiley and Sons (1969) 29-34.
//
//    ****************************************************************
//
//                    VARIABLE DESCRIPTION
//                    --------------------
//
//    *IDEG  = Degree of interpolating polynomial (1 is linear, 2 is
//          quadratic, etc).
//     IERR  = Error code. (0=OK, 1=input variables out of range)
//    *NPTS  = The number of data points in x and y.
//    *X     = The array of independent variable data points.
//    *XARG  = The argument for which an interpolated value is desired.
//    *Y     = The array of dependent variable data points.
//
//     N,N1,L = Dummy variables.
//     I,J = Loop variables.
//     (* Indiates input variables)
//
//    ****************************************************************
//
      double FLAGR (double X[], double Y[], double XARG, int IDEG,
                              int NPTS, int IERR){

      int MAX, NN, N1;
      double FLAGR;
//
      IERR=0;
      FLAGR=0;
      if (IDEG < 1){
                //cout<<"FLAGR:  Illegal value input for IDEG"<<endl;
                IERR=1;}

      if (NPTS < 3){
```

```
                    //cout<<"FLAGR:  Illegal value input for NPTS"<<endl;
                    IERR=1;}

      if (IERR == 1){
            FLAGR=0.;
            return FLAGR;
      }
//
//    **********************
//    End of validity check
//    **********************
//
      NN=abs(NPTS);
    N1=IDEG+1;

      if(X[2] > X[1]){
//
//    ------------------------------------------------------------
//    Check to be sure that XARG is within range of X(I) values
//    for interpolation purposes. If it is not, set FLAGR equal
//    to the appropriate terminal value (Y(1) or Y(N)) and return.
//    Note that this precludes extrapolation of data.
//    ------------------------------------------------------------

            if (XARG <= X[1]){
                  FLAGR=Y[1];
                  return FLAGR;}
       else
                  if (XARG >= X[NN]){
                     FLAGR=Y[NN];
                           return FLAGR;}
                  else
//    -------------------------------------------
//    Data are in order of increasing value of x.
//    -------------------------------------------
                           for (int i=0; i < NN-N1+1; ++i){
                                 int MAX=i+N1;
```

```
                                        if (XARG < X[MAX])
                                                FLAGR=YEST(X,Y,MAX,IDEG,XARG);}
      }
      else
            if (XARG >= X[1]){
         FLAGR=Y[1];
                  return FLAGR;}
       else
                  if (XARG <= X[NN]){
           FLAGR=Y[NN];
                    return FLAGR;}
                  else
//
//     ------------------------------------------
//     Date are in order of decreasing value of x.
//     ------------------------------------------
//
                              for (int i=0; i < NN-N1+1; ++i){
                                        MAX=i+N1;
                                        if (XARG > X[MAX])
                                                FLAGR=YEST(X,Y,MAX,IDEG,XARG);}

return FLAGR;
      }
//
      double YEST(double x[],double y[], int MAX, int IDEG, double XARG){
//
            int MIN;
            double FACTOR, TERM, YEST;

            MIN=MAX-IDEG;
            FACTOR=1.;
            for(int i=0; i < MAX-MIN+1; ++i) {
                  int ic=i+MIN;
                  if (XARG == x[ic]){
                        YEST=y[ic];
                        return YEST;}
```

```
                    FACTOR=FACTOR*(XARG-x[ic]);
            }
            YEST=0;
            for (i=0; i <MAX-MIN+1; ++i){
                    int ic=i+MIN;
                    TERM=y[ic]*FACTOR/(XARG-x[ic]);

                    for (int j=0; j < MAX-MIN+1; ++j){
                            int jc=j+MIN;
                            if (ic != jc)
                                    TERM=TERM/(x[ic]-x[jc]);}
                    YEST=YEST+TERM;
            }
return YEST;
    }


//  ------------------Opens the well ---------------------------------
void open_well(float TP){
    cout<<"Opening Well for "<< TP*HRS<<" miliseconds"<<endl;
}


// -------------------Closes the well ---------------------------------
void close_well(float DT){
    cout<<"Closing Well for "<< DT*HRS<<" miliseconds"<<endl;
}
```

B. Input Data String

```
TID,TOD,CID,D,WP,PTMIN
SGL,SGG,VPAV,TAV,GLR,VISL
EPSD
PIN,PERM,PHI,CF,CWS,SK,RW,RE,H
PMIN,PMAX,NPRES,N
```

C. Example Input Data File:

```
1.995 2.375 4.09 6318. 8.0 29.9
0.834 0.72 15.4 100. 6365. 0.82
0.00003
270. 12. 0.13 0.000002 0.00000047 2.5 0.25 840.34 77.
29. 270. 40 16
```